目 录

第 1 章 遗传算法的基础

1.1 引言

自 1960 年以来，人们对于模拟生物以及由此开发的针对复杂优化问题的有效算法产生了浓厚兴趣。当前在该领域中常常引用的术语就是进化计算(evolutionary computation)。它包含以下一些主要算法：遗传算法(genetic algorithms)(由 Holland 开发[303])，进化策略(evolution strategies)(由 Rechenberg[530]和 Schwefel 开发[567])，进化规划(evolutionary programming)(由 Fogel 等人开发[200])和遗传程序设计(genetic programming)(由 Koza 开发[375])。当然还存在若干将上述算法的各种特点加以结合而形成的混合算法。当前进化计算领域的最新发展水平在 Bäck 和 Schwefel[33]，Michalewicz[454] 以及 Fogel[198] 等人的综述里有很好的介绍。

作为强有力且应用广泛的随机搜索和优化方法，遗传算法可能是当今影响最广泛的进化计算方法之一。在过去的几年中，遗传算法界将更多的注意力放在工业工程领域的优化问题上，并由此产生了一批新的研究和应用[28,198,219,239,249,455,567]。有关遗传算法的参考书目请参阅 Alander 的著述[11]。

一般认为，遗传算法有 5 个基本组成部分(这是由 Michalewicz 归纳的[455])：

1. 问题的解的遗传表示
2. 创建解的初始种群的方法
3. 根据个体适应值对其进行优劣判定的评价函数
4. 用来改变复制过程中产生的子个体遗传组成的遗传算子
5. 遗传算法的参数值

遗传算法维持由一群个体组成的种群 $P(t)$(t 代表遗传代数)。每一个体均代表问题的一个潜在的解。每一个体都被评价优劣并得到其适应值。某些个体要经历称作遗传操作的随机变换，由此生产新的个体。主要有两种变换方法：变异(mutation)的方法是将一个个体改变从而获得新的个体；杂交(crossover)的是方法将两个个体的有关部分组合起来形成新的个体。新产生的个体(称作后代(offspring)$C(t)$)继续被评价优劣。从父代种群和子代种群中选择比较优秀的个体就形成了新的种群。在若干代以后，算法收敛到一个最优个体，该个体很有可能代表着问题的最优或次优解。遗传算法的一般结构可以描述如下：

遗传算法过程

```
begin
    t←0
    初始化 P(t)
    评价 P(t)
    while(终止条件不满足)do
    begin
        重组 P(t)以产生 C(t)
        评价 C(t)
        从 P(t)和 C(t)中选择 P(t+1)
        t←t+1
    end
end
```

关于搜索策略存在两种重要方案：深度搜索最优解和广度搜索解空间[71]。遗传算法提供了一种在复杂解空间上进行有向随机搜索的方法。遗传算子原则上进行的是盲搜索；选择算子则有可能将遗传搜索的方向引导到解空间的理想区域中。针对特定现实世界中问题开发的遗传算法需要注意这样一条普遍原则，即要在对解空间进行深度搜索和广度搜索中维持很好的平衡。为实现这一原则，必须仔细考虑遗传算法的所有组成部分。另外可能还需要结合附加的启发式方法来增强其性能。

1.1.1 编码问题

如何将问题的解编码成为染色体是遗传算法使用中的关键问题。该问题已经从多方面进行过研究，比如当个体需要解码成为解时从基因型空间到表现型空间的映射性质，以及个体被遗传算子操作时的变形特性等。

编码的分类 在 Holland 的工作中，编码采用了二进制字符串(binary strings)的形式[303]。已经知道，由于 Hamming 悬崖的存在，二进制编码对于函数优化问题存在严重缺陷。Hamming 悬崖指的是表现型空间中距离很小的个体对可能有很大的 Hamming 距离[427]。举例来说，个体对 01111111111 和 10000000000 属于表现型空间中的相邻点(最小 Euclidean 距离点)，但它们却在基因型空间具有最大的 Hamming 距离。为了翻越 Hamming 悬崖，个体的所有位需要同时进行改变。由杂交和变异实现翻越 Hamming 悬崖的可能性非常小。在这种情况下，二进制编码无法维持表现型空间中点的位置。

对于工业工程领域里的许多问题而言，几乎不可能用二进制编码来表示它们的解。在过去的 10 年里，已经针对特定的问题提出了各种编码方法，其目的都是为了能够更有效地实现遗传算法。根据采用何种符号作为基因的等位基因，编码方式可以分类如下：

- 二进制编码(binary encoding)
- 实数编码(real-number encoding)
- 整数或字母排列编码
- 一般数据结构编码

实数编码对于函数优化问题最为有效。关于实数编码在函数优化和约束优化领域比二进制编码和 Gray 编码更有效的说法,已经得到了广泛的验证[180,447,455,647]。由于实数编码基因型空间中的拓扑结构与其表现型空间中的拓扑结构一致,因此很容易从传统优化方法中借鉴好的技巧来形成有效的遗传算子。整数和字母排列编码(literal permutation encoding)对于组合优化问题最为有效。由于组合优化问题最关键的是要寻找满足约束项目的最佳排列或组合,因此字母排列编码对于这类问题是最有效的方法。对于更为复杂的现实问题,用合适的数据结构来表示基因的等位基因,可以有效抓住问题的本质。在这种情况下,基因可能是 n 维数组或更为复杂的数据结构。

根据编码的结构,编码方法还可以分为如下两类:(1)一维编码(one-dimensional encoding),(2)多维编码(multidimensional encoding)。大多数实践中采用了一维编码。然而许多实际问题需要多维结构的解,用多维编码方法来表示这些解就很自然。比如,Vignaus 和 Michalewicz 对运输问题采用了分配矩阵进行编码[641]。Cohoon 和 Paris 对 VLSI 电路放置问题采用了二维编码[127]。Anderson, Jones 和 Ryan 采用了二维网格型编码[18]。Moon 和 Kim 对于图问题采用二维编码[462]。Ono, Yamamura 和 Kobayashi 对于作业车间调度问题采用了作业顺序矩阵编码[493]。Bui 和 Moon 给出了关于多维编码和杂交的一般性讨论[79]。他们在文中指出,将多维问题的解进行一维编码必然会损失多维结构中相当数量的信息。

根据编码的内容,编码方法还可看作如下两类:(1)仅包含解,(2)包含解和参数。在遗传算法实践中,第一种方法被广泛用来针对给定的问题开发合适的编码。第二种方法在 Rechenberg 和 Schwefel 提出的进化策略中被采用[567]。一个个体包含两个部分:首先是给定问题的解,其次是策略参数,包括变异中正态分布的方差和协方差。将策略参数并入个体表示的目的,是通过将进化算子应用于这些参数来促进它们的进化自适应。因此搜索就在解空间和进化参数上同时进行。通过这种方法,可以在任意环境下获得变异参数的合理调整和多样性。

不可行(infeasibility)与非法性(illegality)　遗传算法交替地在编码空间和解空间中进行操作。换句话说,也就是交替地在基因型空间和表现型空间中进行操作。遗传算子作用于基因型空间中,而评价和选择则作用于表现型空间中。自然选择连接了染色体和解码产生的解的性能。从基因型空间到表现型空间的映射对于遗传算子的性能有很大影响。其中一个与映射相关的重要问题就是某些个体对应着给定问题的不可行解。对于约束优化问题和组合优化问题而言,这个问题可能很严重。

我们必须明确区分两个基本的概念：不可行和非法性(图 1.1)。这两个概念经常在文献中被错误使用。不可行指的是某个染色体解码出来的解在给定问题的可行区域之外的现象；而非法性指的是某个染色体不能代表给定问题的解的现象。

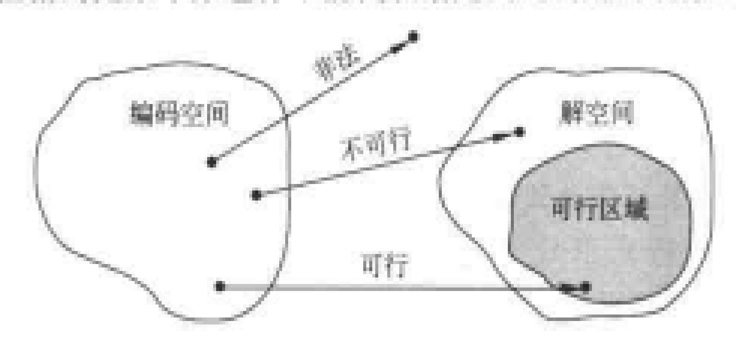


图 1.1 不可行和非法性

染色体的不可行起源于约束优化问题。无论是采用传统方法还是遗传算法，都必须处理约束。对于许多优化问题来说，可行区域可以表示为由等式和不等式组成的系统。在这种情况下，罚方法可以用来处理不可行染色体[218,455]。在约束优化问题中，最优解常常出现在可行与不可行区域的交界处。罚方法可以迫使遗传搜索分别从可行和不可行区域两方面逼近最优解。

染色体的非法性起源于编码方式。对于许多组合优化问题来说，通常采用的是依赖于问题的编码方式。这种编码当采用简单单点杂交操作时通常会产生非法后代。由于非法染色体不能被解码为一个解，罚方法无法处理这种情况。修补方法通常用来将非法染色体转换为合法。举例来说，著名的 PMX 算子从本质上来说就是对于排列表示的两点杂交，同时带有修补过程来消除由简单两点杂交导致的非法性问题。Orvosh 和 Davis[494]的研究表明，对于许多组合优化问题来说，修补不可行或非法的染色体相对比较容易，而且修补策略的效果比其他诸如抛弃法或罚方法要好。

编码性质 每当提出一种新的编码方式，就需要验证能否采用该编码建立有效的遗传搜索。关于评价编码方式的原则有如下几个方面[370,536]：

- 不冗余
- 合法性
- 完备性
- Lamarckian 性质
- 因果性

性质 1.1(不冗余(nonredundancy)) 从编码到解的映射应该是 1 对 1 的。

从编码到解的映射可能是下列 3 种情况之一(图 1.2)：

1. 1 对 1 的映射
2. n 对 1 的映射

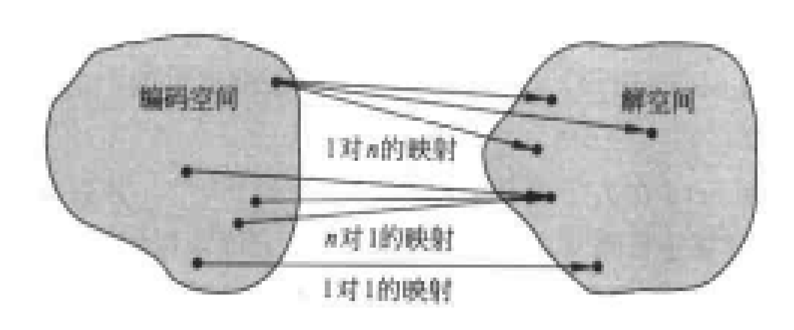


图 1.2 从染色体到解的映射

3. 1 对 n 的映射

最希望的情况是 1 对 1 映射，该映射确保在产生后代时不会进行无价值的操作。如果是 n 对 1 的映射，遗传算法在搜索过程中会浪费时间。在这种情况下，由于两个个体在表现型空间中一样，但在基因型空间中不一样，因此在基因型空间中的距离度量无法将这两个个体处理为相同个体。n 对 1 的映射于是就成为遗传算法早熟收敛的原因之一。最不希望产生的是 1 对 n 的映射，原因是还需要其他的方法在表现型空间中从许多可能的解中确定一个。

性质 1.2（合法性（legality）） 对编码的任意排列都对应着一个解。

该性质确保大多数现有的遗传操作可以被简单地应用于该编码。

性质 1.3（完备性（completeness）） 任意解都对应着一个编码。

该性质确保解空间中任意点对于遗传搜索来说都是可达的。

性质 1.4（Lamarckian 性质（Lamarckian property）） 某个基因的等位基因的含义不依赖于其他基因。

关于编码的 Lamarckian 性质考虑的是一个染色体能否将其价值通过常用的遗传操作传到将来的种群中的问题[114]。现在看一个例子。9 城市旅行推销员巡回问题可以采用如下的 Bean 随机键方法来编码[47]：

随机键：[0.34　0.09　0.88　0.66　0.91　0.01　0.23　0.21　0.44]

可以将键按照升序排列来获得一个巡回：(6－2－8－7－1－9－4－3－5)。考虑子巡回[8　7　1　9]。用随机键表示的该子巡回为[0.88　0.66　0.91　0.01]。在后代中该子随机键通常表示为另一个子巡回，而不是子巡回[8　7　1　9]。该子随机键表示的子巡回需要由子代染色体中其他基因的值来确定。事实上，后代除了随机机会以外，没有从父代获得任何关于有效子巡回的信息。这种编码称之为非 Lamarckian 的，即某个基因的等位基因的含义由依赖于上下文的方式来确定。这样，后代通常从父代中不继承任何东西。一般来说，我们希望编码方式具有 Lamarckian 性质，即后代能够从其父代继承优秀的品质。在半 Lamarckian 类型中，染色体中的某些部分从父代中继承相同的内容，而

另一些部分的含义需要与其他基因一同确定。

性质 1.5(因果性(causality)) 由于变异带来的基因型空间中的小扰动在表现型空间中也对应着小扰动。

该性质由 Rechenberg 在关于进化策略的讨论中提出[530]。该性质强调保护邻域结构。也就是说,在变异算子带来有用的新信息的同时,它应该保留在对应表现型空间中的邻域结构。这种邻域结构保留的观点在遗传优化的研究人员中是一种常识[179]。不破坏邻域结构的搜索过程表现出强因果性。与之相对的极端情况是无因果性。弱因果性描述的是在基因型空间中小的扰动对应着表现型空间中大的变化的情况,反之亦然。Sendhoff, Kreuts 和 Seelen 提出了一个条件来度量与变异算子相结合的从基因型到表现型映射中的因果性[571]。

1.1.2 遗传算子

当很难事先获得问题的求解步骤时,搜索就成为更为广泛的问题求解方法之一了。有两种有代表性的搜索行为:随机搜索和局部搜索。*随机搜索*(random search)广泛探索整个解空间并且能够从局部最优中逃离。*局部搜索*(local search)深度探索最优解并且能够向着局部最优解进行爬山。这两种搜索能力构成了整个搜索彼此互补的组成部分。理想的搜索应该同时具有这两种搜索性质。采用传统方法来设计这样的理想搜索方法几乎是不可能的。遗传算法是一种结合了有向和随机两种能力的通用搜索方法。该算法可以在对搜索空间进行深度搜索和广度搜索之间维持很好的平衡。在遗传算法中,由选择机制来深度搜索积累的信息,由遗传算子来广度搜索解空间中新的区域。

在传统遗传算法中,杂交算子被用作主要的算子,遗传系统的性能在很大程度上依赖于杂交算子的性能。变异算子在各个染色体中自动产生随机的变化,通常被用作次要的算子。从本质上讲,遗传算子执行的是随机搜索,不能保证产生改进的后代。已经发现对于许多大规模组合优化问题,遗传算法的收敛速度很慢。对于杂交和变异有许多经验式的比较研究。关于变异有时比杂交更重要的论断已经得到确认。

关于解释遗传算法是如何搜索分布式信息来产生优秀个体存在两个假设:(1)*积木块*(building-block)假设,(2)受控收敛偏差假设。积木块假设由 Holland 提出[303],并由 Goldberg 加以改进[249]。根据该假设,杂交算子重组两个父代的特性来产生子代。有时杂交会重组两个父代的最好特性并产生优秀的子代。由于个体的适应值通常依赖于简单性质的复杂模式,因此算子能够把对父代适应值有所贡献的这些模式传播给子代就显得非常重要。*异位显性*(epistasis)的概念指的就是编码中基因间的强相互作用。换句话说,异位显性度量的是一个基因对适应值的贡献依赖于其他基因的程度的大小。对于一个给定的问题,高度的异位显性意味着无法形成积木块。

受控收敛偏差假设由 Eshelman, Mathias 和 Schaffer 提出[178]。该假设认为应利用

种群收敛的程度来指导搜索。新的点从某一分布中进行采样，该分布是种群任一时刻分布的函数。随着种群逐渐收敛，偏差变得越来越小。积木块假设强调在父代种群中保留的性质的重组与传播，而受控收敛假设则强调从当前种群的分布函数中进行随机采样。

如何概念化遗传搜索将对遗传算子的设计产生影响。从搜索能力的角度出发，期望设计出能够同时具有随机搜索和有向搜索两种能力的方法。Cheng 和 Gen 提出了下面的设计遗传算子的方法[105]。对于两个遗传算子杂交和变异来说，一个用来执行随机搜索并试图探索局部最优解以外的区域，而另一个则用来执行局部搜索并试图导找到更好的解。遗传搜索于是具备了两种搜索能力。采用这种方法，变异算子将与杂交算子在遗传搜索中具有同等重要的地位。

1.1.3 选择

遗传算法的原理从本质上来说基于达尔文的自然选择学说。选择提供了遗传算法的驱动力。如果驱动力太大，遗传搜索将过早地终止；而如果驱动力太小，进化过程将慢得难以接受。通常需要在遗传搜索的早期采用较小的选择压力（用来对搜索空间进行广度探索），而在晚期则推荐采用较大的选择压力（用来限制搜索空间）。选择将遗传搜索引导到搜索空间中有前途的区域中。在过去的 20 年中提出、验证并比较了许多选择方法。通常采用的选择方法如下：

- 轮盘赌选择(roulette wheel selection)
- $(\mu+\lambda)$选择($(\mu+\lambda)$selection)
- 竞争选择(tournament selection)
- 稳态复制(steady-state reproduction)
- 排序与比例变换
- 共享(sharing)

Holland 提出的轮盘赌选择是最知名的选择方式[303]，其基本的原理是根据每个染色体适应值的比例来确定该个体的选择概率或生存概率。因此可以建立一个轮盘赌模型来表示这些概率。选择的过程就是旋转轮盘若干次（次数等于种群规模），每次为新种群选出一个个体。轮盘这种选择方法的特点就是随机采样过程。Baker 提出了仅采用一次轮盘旋转的随机通用采样(stochastic universal sampling)方法[36]。该方法采用了与标准轮盘赌一样的轮盘，区别在于该方法采用了均匀分布的且个数等于种群规模的旋转指针。这种方法的基本原则是保证每个染色体在下一代中复制的次数与其期望值相差不大。

与比例选择不同，Bäck 提出了$(\mu+\lambda)$选择和(μ,λ)选择。该方法为确定性选择过程，从父代和子代中选取最好的染色体[27]。值得注意的是该方法禁止从种群中选取相同染色体，因此许多研究人员倾向于采用该选择方式来处理组合优化问题。截断选择

(truncation selection)和区段选择(block selection)也是确定性选择过程[619]。这两种方法根据适应值对个体进行排序并从中选出最好的作为父代。最优性选择(elitist selection)一般作为比例选择方式的补充,它可以确保最优秀的染色体在不被比例选择选中的情况下能够在新一代中得以保留。

世代替换(generational replacement)将所有的父代用其子代替换,因此可以看作是另一种方式的确定性选择。由 Whitley[662]和 Syswerda[603]提出的稳态复制就属于这种类型。在这种方法中,n 个最差的父代被子代替换(n 是子代的个数)。

另一类选择方式同时包含随机和确定性的特征。其中最有代表性的就是由 Goldberg,Korb 和 Deb 提出的竞争选择(tournament selection)[251]。该方法随机选择一些染色体并从中选出最好的来进行繁殖。选取的染色体数量称作竞争规模(tournament size)。常用的竞争规模为 2,这种情况被称作二进制竞争(binary tournament)。Wetzel 提出了随机竞争选择[659]。这种方法采用普通的方法计算选择概率,然后采用轮盘赌相继选出染色体对。在选出染色体对后,其中具有高适应值的个体进入新种群。该方法一直进行到新种群的个体数量达到种群规模为止。由 Brindle 提出的剩余随机采样(remainder stochastic sampling)是他提出的确定性采样的改进方法[76]。这种方法首先根据每个染色体被复制个数的期望值的整数部分,选出若干个新个体,然后根据期望值的小数部分进行种群中余下位置的竞争。

在比例选择过程(proportional selection procedure)中,个体的选择概率与其适应值大小成正比。这种简单的选择方法存在一些我们不希望的性质。举例来说,在算法进行的早期,个别超级染色体具有控制选择过程的趋势;而在算法进行的晚期(这时种群已经大部分收敛),个体间的竞争并不激烈,呈现出随机搜索的行为。

比例变换和排序机制就是用来解决这些问题的。比例变换方法将原始目标函数值映射为正实数,每个染色体的生存概率就根据这些正实数进行计算。比例变换带有双重目的:(1)维持染色体相对适应值比例间的合理差异;(2)在算法早期通过抑制某些超级染色体过快收敛来达到限制竞争的目的,在算法晚期则加速竞争。

从 De Jong 开展这方面的工作以来,使用比例变换目标函数已经被广为接受,提出了若干种比例变换方法。根据将原始适应值变换到比例变换适应值的函数类型不同,可以分为线性比例变换[270]、Sigma 截断[203]、幂比例变换[245]、对数比例变换[270]等。如果从原始适应值到比例变换适应值的转换关系是固定的,这种方法称作静态比例变换方法(static scaling method);如果转换关系是根据某些参数变化的,这种方法称作动态比例变换方法(dynamic scaling method)。窗方法在适应值比例选择中采用了移动的基准来维持恒定的选择压力[281]。标准化方法也是一种由 Cheng 和 Gen 提出的动态比例变换方法[105]。

对于大多数比例变换方法而言,其参数的选取由问题决定。适应值排序具有类似于适

应值比例变换的效果,同时还避免了比例变换参数的选取[531]。Backer 为了克服比例变换参数选取的问题而提出了直接基于适应值的遗传算法排序选择方法[35,250,281]。排序方法忽略了实际的目标函数值,与之相对应的是采用了染色体的排序来确定其生存概率。这种方法的思路很直观:对种群进行从好到差的排序,然后根据每个染色体的排序(而不是原始适应值)确定其选择概率。常用的有两种方法:线性排序和指数排序。

由 Goldberg 和 Richardson 针对多峰函数优化提出的共享方法[248]可以用来维持种群多样性。共享函数就是一种用于根据个体在某个距离内与其他个体相邻的程度来确定该个体适应值下降多少的方法。由于适应值的下降,在拥挤的峰周围的个体的复制概率受到抑制,而其他个体则易于产生后代。

1.1.4 遗传局部搜索

在过去的 10 年里,人们已经详细研究了将遗传算法和局部搜索启发式方法相结合从而解决优化问题的想法,并提出了多种结合方法。一种最常见的混合遗传算法是在规范遗传算法中将局部优化作为其辅助成分。在混合算法中,每一个新产生的后代在进入种群之前需要进行局部优化使其移动到局部最优点上。遗传搜索进行种群中的全局广度搜索;而局部搜索则进行染色体中的局部深度搜索。由于遗传算法和局部搜索方法的互补特性,混合方法通常比任何一种方法的效果都要好。

存在两种常见的遗传局部搜索形式。一种以 Lamarckian 进化为特色,另一种以 Baldwin 效应为特色[663]。两种方法都采用了个体在整个生命过程(一代)中都进行学习(爬山)这样的比喻说法。在 Lamarckian 进化中,个体爬山后被重新放回到种群中。而在 Baldwin 效应中,个体的基因型保持不变,改变的仅为其适应值。根据 Whitley, Gordon 和 Mathias 通过一些测试问题获得的经验,在采用相同局部搜索方法的前提下,存在 Lamarckian 策略收敛到局部最优解而 Baldwinian 搜索策略收敛到全局最优解的情况。然而在所有的测试问题中,Baldwinian 策略均比 Lamarckian 要慢许多。

将遗传算法与 Lamarckian 进化原理相结合的早期工作综述如下。Grefenstette 将 Lamarckian 算子引入遗传算法[269]。Davidor 定义了变异操作的 Lamarckian 概率,目的是使得变异算子更为可控,同时还为遗传算法引入一些高质量的局部爬山算子[141]。Shaefer 为标准遗传算法(从本质上来说就是 Lamarck 的)添加了在染色体与解空间之间的中间映射[572]。

Kennedy 为结合了 Lamarckian 进化原理的混合遗传算法提供了一种解释[353,663]。由 Holland 提出的规范遗传算法从 Darwin 的自然选择原理获得灵感。19 世纪,Darwin 的理论受到 Lamarck 的挑战。Lamarck 认为由环境变化引起的生物体生命过程中的结构变化可以传递到其后代中去。这种理论认为生物体可以将他们在生命过程中获得的知识和经验传递给后代。虽然现今的生物学家均认为从自然世界获得的特性无法继承,但 Lamarckian 理论

的影响力还是在社会进化中得以体现。通过结构化的语言和文化，观念和知识得以代代相传。遗传算法作为人工生物体，可以利用Lamarckian理论的优势。通过让某些个体的经验得以传递到未来的个体中去，我们可以将遗传算法的搜索领域集中到最有希望的区域，从而改进算法的性能。采用了Lamarckian方法，传统的爬山程序可以采用后代作为初始点并对其进行快速的局部优化。个体学会在局部解空间中爬山后，其后代就可以经历遗传算法的评价和选择阶段。后代可以利用普通的杂交将其经验传递到后代中去。

用$P(t)$和$C(t)$代表当前遗传代数t的父代和子代。混合遗传算法的一般结构可以表示如下：

混合遗传算法过程

```
begin
    t←0
    初始化 P(t)
    评价 P(t)
    while 终止条件不满足 do
    begin
        重组 P(t)以产生 C(t)
        对 C(t)进行局部爬山
        评价 C(t)
        从 P(t)和 C(t)中选择 P(t+1)
        t←t+1
    end
end
```

在混合方法中，人工生物体首先经历Darwin的生物进化，然后经历Lamarckian的智能进化(图1.3)。在Lamarckian进化中在评价过程之前，用传统的爬山程序来将某种"知识"注入到后代生物体中。

Moscato和Norman引入了memetic算法(memetic algorithm)这一术语，用于描述局部搜索起重要作用的遗传算法[468]。该术语来源于Dawkin将meme看作可在人们交换想法时进行自我复制的信息的单位[140]。在基因与meme间存在本质的区别。在meme进行转递之前，它通常被发送者转换为它对meme的思考、理解和处理，而基因则进行整体的传递。Moscato和Norman将这种想法与局部改善联系起来，由此提出了术语memetic算法用以描述局部搜索起重要作用的遗传算法。

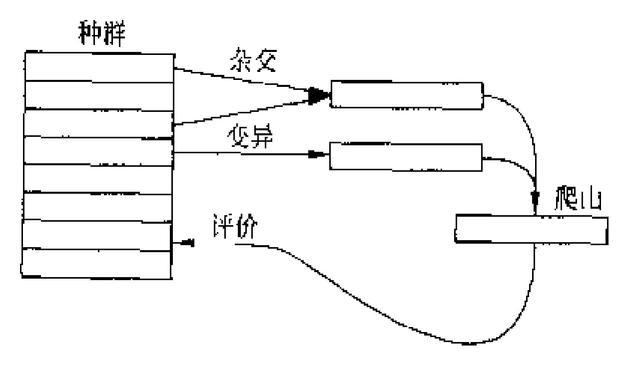


图1.3 混合遗传算法的一般结构

Radcliffe 和 Surry 给出了对 memetic 算法的正式描述[520]，该描述为考虑 memetic 和遗传算法提供了类似的正式框架。根据 Radcliffe 和 Surry 的观点，如果某个局部优化方法与遗传算法相结合并且在所有个体放入种群之前均要执行该算法，memetic 算法可以被简单地看作一种在局部最优子空间上进行的特定遗传搜索。重组和变异操作通常将产生局部最优子空间以外的解，而局部优化算法可以对这些解进行修补并产生在子空间内部的最终后代，这样就产生了 memetic 算法（如图 1.4 所示）。

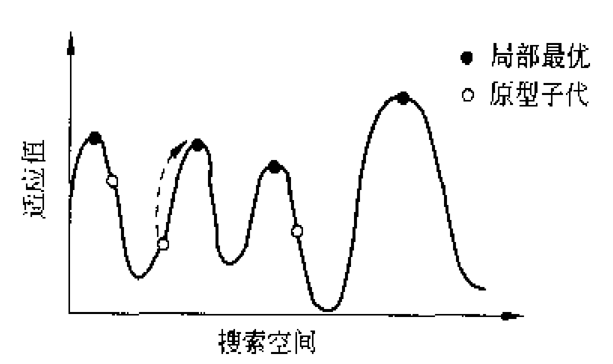


图 1.4 memetic 算法和局部搜索算法

（改编自 Radcliffe 和 Surry[520]）

许多研究人员对局部搜索在遗传算法环境中的作用进行了严肃认真的思考，许多成功的应用明显地说明了混合方法的优势。mememtic 算法和 Lamarckian 进化都试图给出基于不同自然现象的混合方法的合理解释。

1.2 遗传算法的适应性

鉴于遗传算法是受进化思想启发而来的，人们很自然希望适应性不仅用于寻找给定问题的解，同时还可以让遗传算法针对特定问题进行变化。在过去的几年里，为了将遗传算法针对现实世界的问题有效地实现，已经提出并测试了许多适应性(adaptation)方案。总起来说，存在两类适应性：

1. 对问题的适应性
2. 对进化过程的适应性

以上二者的区别在于：前者提出修改遗传算法的某些元素（比如表示、杂交、变异和选择）从而得到算法的理想形式以满足给定问题的本质；后者提出一种在遗传算法解决问题的同时对其参数进行动态调整的方法。根据 Herrera 和 Lozano 的报道，后者的适应性可以进一步分为下列几类[294]：

- 适应性参数设置
- 适应性遗传算子
- 适应性选择

- 适应性表示
- 适应性适应值函数

在上述这些类别中，参数适应性(parameter adaptation)在过去的10年中得到了比较充分的研究，原因在于诸如变异率、杂交率和种群规模等策略性参数对于确定深度搜索和广度搜索的折中是关键的因素。长久以来，人们已经得到了一致的认识，即这些策略性参数对于遗传算法的性能有显著的影响[146,268]。

1.2.1 结构适应性

遗传算法是以二进制编码和二进制遗传算子为特征而创建的普适但搜索能力不如特定算法强的方法。这种方法需要将原始问题修改为适合遗传算法的形式，如图1.5所示。该方法包括从可能解到二进制表示的映射，对解码和修补过程的考虑等方面。对于复杂问题，这样的方法通常无法提供有效的解答。

为了解决上述问题，已经针对特定的问题提出了若干种遗传算法的非标准实现方式。这种方法不改变问题的形式，而是通过修改遗传算法对可能解的染色体表示和采用合适的遗传算子来适应问题，如图1.6所示。

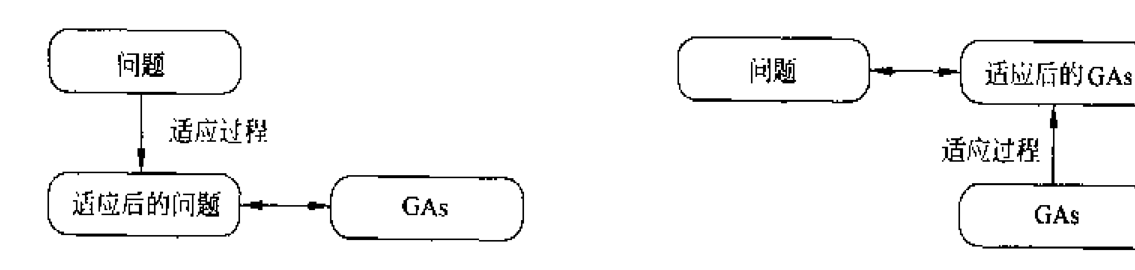


图1.5 让问题来适应遗传算法

图1.6 让遗传算法来适应问题

但是一般来说，选用给定问题的整个原始解来作为染色体不是好的方法。原因在于许多实际问题太复杂，无法找到遗传算法对完整解表示的合理实现方式。通常编码方式可以是直接的或非直接的。在直接编码中，将给定问题的完整解作为染色体。然而对于复杂问题来说，这种方法会使得大量后代不可行或非法，因此几乎所有传统的遗传算子都不可用。与之相对的非直接编码仅将解的必要部分作为染色体。可以利用解码过程由染色体来产生解。解码过程是依赖于问题的过程，用于根据由遗传算法产生的排列项和(或)组合项生成问题的解。采用这种方法时，遗传算法仅在感兴趣的解空间上进行搜索。

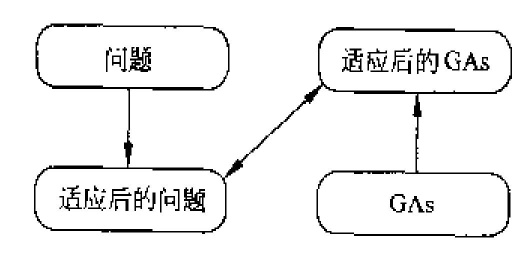


图1.7 对遗传算法和问题均进行适应性调整

第三种方法是对遗传算法和问题均进行适应性调整，如图1.7所示。组合优化问题的一个共同的特征是寻找满足许多边界约束的排列和(或)组

合项。如果能够确定排列和(或)组合项,就可以采用依赖于问题的方法来导出问题的解。采用第三种方法,遗传算法用于进化出所考虑项的合理排列和(或)组合,启发式方法随后用来根据排列和(或)组合项构造出问题的解。这种方法被成功地应用于工业工程领域,并且最近成为遗传算法实际应用的主要方法[291,455]。

1.2.2 参数适应性

遗传算法的性能由在搜索空间进行的深度搜索(exploitation)和广度搜索(exploration)的平衡决定。这种平衡深受诸如种群规模、最大遗传代数、杂交率和变异率等策略性参数的影响。如何设置参数值以及如何有效地寻找参数值都是遗传算法领域中重要并且有前景的研究方向。最近关于适应性方法技巧的综述参见 Herrera 和 Lozano[294]以及 Hinterding, Michalewicz 和 Eiben[299]的介绍。

大多数遗传算法的应用采用的都是固定参数。参数值的选取采用的是设置-测试的方法。由于遗传算法从本质上讲是动态和适应性的过程,因此采用固定参数的方法是与一般的进化精神相违背的。因此很自然地希望能够在算法运行的过程中修改策略性参数的值。这种想法可能由如下的方法来实现:(1)通过采用一种规则,(2)通过获取当前搜索状态的反馈,(3)采用自适应机制。根据 Hinterding,Michalewicz 和 Eiben 对适应性的分类,有 3 种原则性的分类:(1)确定性的,(2)有适应能力的,(3)自适应的。

确定性的适应性(deterministic adaptation) 如果策略性参数的值由确定性规则来修改,就称作确定性的适应性。通常采用根据遗传代数来进行改变的时变性策略。比如,随着遗传代数的增长,变异率逐渐下降,如下式所示:

$$p_m = 0.5 - 0.3\frac{t}{G}$$

其中 t 是当前遗传代数,G 是最大代数。随着遗传代数增长为 G,变异率会从 0.5 下降为 0.2。依赖于时间的变异率首先由 Holland 提出,虽然他并没有给出详细的参数选择方案[303]。这种方法早期的例子包括 Fogarty[193]和 Hesser 与 Männer[296]的工作。

有适应能力的适应性(adaptive adaptation) 如果从进化过程中获取某种形式的反馈并用来确定策略性参数改变的方向和(或)大小,就称作有适应能力的适应性。这种方法早期的例于包括 Rechenberg 在进化策略中的 1/5 成功准则,用于修改变异的步长[530]。该准则认为,所有变异中成功的变异比例为 1/5。因此,如果该比例大子 1/5,则步长应该增加,而如果该比例小于 1/5,则步长应该减小。Davis 提出的适应性算子适应值利用成功进行复制的比较大的算子数目来调整算法的参数比率[147]。Julstrom 的适应性机制根据杂交算子和变异算子的性能来调节其参数比率[341]。关于这种类型的学习规则机制的深入研究由 Tuson 和 Ross 完成[635]。

自适应的适应性(self-adaptive adaptation) 自适应的适应性允许策略性参数在进

化过程中自行进化。这些参数也进行编码并经历变异和重组。编码的参数并不直接影响个体的适应值，但更优秀的参数值将产生更优秀的个体。这些个体很可能在选择过程中生存并产生后代，由此传播更好的参数值。进行自适应调整的参数可以是那些控制遗传算法操作的、控制复制和其他算子操作的、或者进行替代搜索过程的概率。Schwefel 在进化策略中开发了这种方法，用于变异步长和变异旋转角的自适应调整[567]。Hinterding 在下料问题中采用了多染色体来实现邻近自适应。这种自适应用于对两种可能的变异算子的选用和整体变异算子的强度进行适应性调整。

1.2.3 模糊逻辑控制器

约在20世纪80年代晚期，模糊控制(fuzzy control)成为模糊集理论应用中最活跃、最多产的领域之一。Lee 给出了该研究领域全面的综述[395,396]。与传统逻辑系统相比，模糊逻辑与人类思考和自然语言的本质更为接近。模糊逻辑提供了一种获取真实世界近似、不精确本质的有效方法。从这种观点看，模糊逻辑控制器(fuzzy logic controller)的基本部分是一系列与模糊含意和合成推理规则这二重概念相关的语义学控制规则。从本质上来说，模糊逻辑控制器提供了一种能够将基于专家知识的语义学控制策略转变为自动控制策略的算法。特别地，当控制过程过于复杂而不能用传统方法进行分析，或者当可得到的信息源被定性、不精确或不确定表示时，这种方法显得非常有效。因此模糊逻辑控制可以看成传统精确数学控制与人或类似人决策过程妥协的重要步骤。

将模糊逻辑方法用于动态调整遗传算法策略性参数的开创性工作，由 Lee 和 Takagi[402] 以及 Xu 和 Vukovich[673] 完成。其主要思想是采用模糊逻辑控制器来估算遗传算法中策略性参数的新数值。控制器的输入是遗传算法性能度量的任意组合以及算法的当前参数，输出是参数值。

根据 Lee 的观点，模糊逻辑控制器由4个主要部分组成[395]：

1. 知识库
2. 模糊化接口
3. 推理系统
4. 解模糊化接口

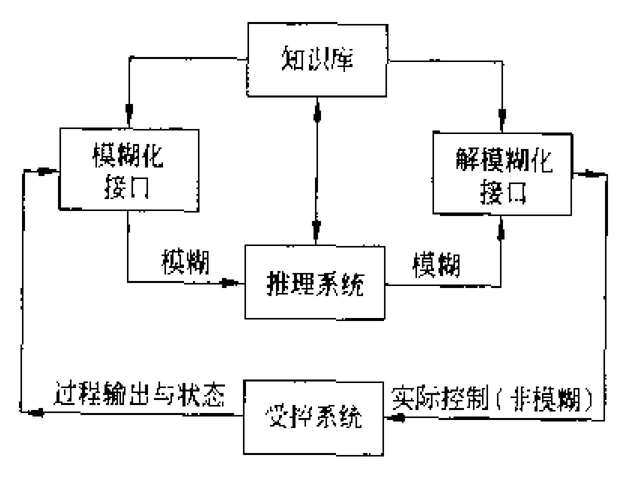


图1.8 模糊逻辑控制器的一般结构
(引自 Lee[395])

专家的知识以语义学规则的方式存放在知识库中。模糊化接口用于将清晰的数据转换为模糊数据。控制器的核心推理系统基于知识库进行近似推理。解模糊化接口将模糊控制行为转为非模糊控制行为。模糊逻辑控制器的一般结构见图1.8。

模糊逻辑控制器是一类建立在模糊逻辑和模糊集理论基础之上的基于规律的系统。参数模型用于实现可由语义学意义指定的输入-输出映射。模糊与非模糊系统的区别在于它们如何划分空间和解决冲突。在模糊系统中，划分可以重复，在输入空间中的1点可以属于多个划分。当采取行动时，该点属于的每一个划分都根据点属于该划分的程度对决策有不同的贡献。这种混合机制给予基于规则的模糊系统平滑的修改行为。

模糊系统的动态行为以基于专家知识的一系列语义学描述规则为特征。专家知识通常具有下面的形式

如果一系列条件满足，则可以推知一系列结果。

由于上述"如果-则"规则的前因后果与模糊概念(语义学术语)相关，通常称它为模糊条件陈述。此外，这些规则的前因后果中还可能包含若干语义学变量。当这种情况发生时，系统被称作多输入/多输出模糊系统。在模糊逻辑控制器中，模糊控制规则就是模糊条件陈述，其中前因就是在应用领域里的条件，后果就是受控系统的控制行为。从根本上来说，模糊控制规则为表示控制策略和领域知识提供了比较方便的形式。

知识库有两个组成部分：数据库和规则库。数据库中包含了在"如果-则"规则的前因后果中使用的语义学变量的定义。这些定义由模糊集的隶属度函数组成。规则库由表达专家知识的模糊控制规则组成。

如果要用模糊控制器来调整遗传算法的策略性参数，对种群多样性的度量、适应值和当前参数值就作为"如果-则"规则的输入。输出则给出了策略性参数(如杂交率、变异率、种群规模、选择压力)的数值或者这些参数的改变值。所有的输入和输出均需要有一系列相关的语义学标签。这些标签的含义由模糊集的隶属度函数来确定。因此所有的输入输出均需要给出其值的边界范围，然后在其中定义隶属度函数。在选择完输入输出并定义好数据库后，就应该确定描述输入输出关系的模糊规则。由两种方法来确定：(1)采用专家的经验与知识，(2)采用自动学习方法。

一般来说，遗传算法的行为受许多不确定因素的影响，同时在认定策略性参数与遗传算法行为之间关系时，仅有不完整的知识和不精确的信息可以获得。因此，采用模糊逻辑控制器来动态调整这些参数就成为可接受的方案了。

Zeng 和 Rabenasolo 的方法 Zeng 和 Rabenasolo 方法的基本结构包含3个模糊逻辑控制器：一个用来控制杂交率，一个用来控制变异率，一个用来控制杂交位置[692]。模糊逻辑控制器用来近似遗传算法策略性参数和种群中各种性能度量之间的关系。这些度量(常用来表示遗传算法的行为)组成了控制器的输入变量，而策略性参数 p_c，p_m，和 p_ω 则用作控制器的输出变量。控制器的模糊规则从作者对若干测试问题进行优化得到的经验中提取。

一些用来优化遗传算法行为的启发式原则用来调整杂交率，这些原则包括：

- 为了维持种群的多样性，较远的个体有较高的机会被选中进行杂交。
- 为了在最优区域中进行搜索，具有较高适应值的两个近距离个体有较高机会被选中进行杂交。
- 为了避免收敛到局部最优，如果适应值的方差很小，则需要加强杂交操作。
- 为了稳定最优种群，如果被选中进行杂交的个体的适应值与当前种群中最大适应值比较接近，则杂交操作需要削弱。

这些原则中有一些明显的冲突。然而通过正确调整这些冲突，就可以获得遗传算法的优秀行为。

假设 $P=\{x_1,x_2,\cdots,x_n\}$ 代表当前种群，其中 n 是种群规模。相应的适应值由 F 表示。遗传算法的目的是寻找最优的 x，使得 $F(x)=\max\{F(x_i)\mid i=1,2,\cdots,n\}$。根据前面的原则，个体对 (x_i,x_k) 之间的杂交概率 p_c 由下面的参数决定：

- 适应值的变化：$\mathrm{var}=(F_{\max}-\overline{F})/(F_{\max}-F_{\min})$
- x_i 的适应值和 $F_{\max}$ 的距离：$G=[F_{\max}-F(x_i)]/(F_{\max}-F_{\min})$
- x_i 和 x_k 之间的距离：$D=d(x_i,x_k)$
- 适应值：$F_1=F(x_j)/F_{\max}$ 和 $F_2=F(x_k)/F_{\max}$

参数 var，F_1，F_2 和 D 均包含在区间[0,1]中。在这些参数中，var 是由整个种群决定的，而其他参数则由特定的个体来确定。在模糊逻辑控制器中，这些参数作为输入变量，而 p_c 的值则作为其产生的输出变量之一。对于每一输入变量，其隶属度函数的定义如图 1.9所示，其中 $input_i\in\{\mathrm{var},G,F_1,F_2\}$ 是控制器的任意输入变量，w_i 是从遗传算法的经验中获取的对应参数。p_c 的隶属度函数定义如图 1.10 所示。

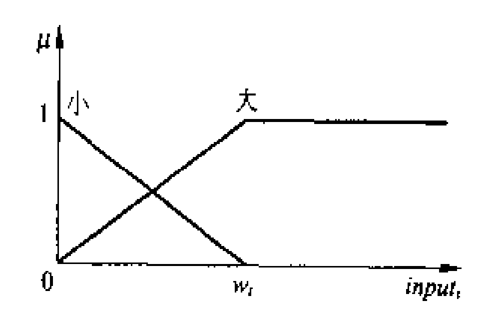


图 1.9 输入变量的隶属度函数定义

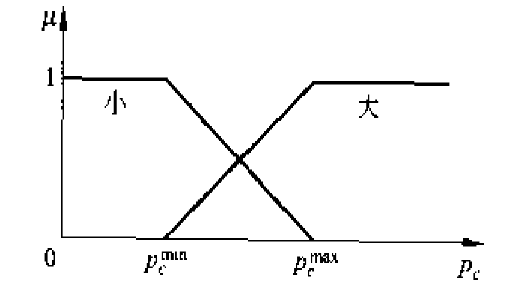


图 1.10 输出变量的隶属度函数定义

采用这样的定义，上述原则可以转换为下面的模糊规则：

1. 如果 G 大，则 p_c 大。
2. 如果 var 小且 G 小，则 p_c 小。
3. 如果 D 小且 F_1 大而且 F_2 大，则 p_c 大。
4. 如果 D 小且(F_1 或 F_2)小，则 p_c 小。
5. 如果 D 大，则 p_c 大。

参数 p_m 的选取原则为：如果遗传算法趋于收敛到局部最优，则增大变异率；如果当前种群多样性很强或者得到了全局最优的种群，则减小变异率。对应的模糊规则包括：

1. 如果 var 小且 G 大，则 p_m 大。
2. 如果 var 小且 G 小，则 p_m 小。
3. 如果 var 大，则 p_m 小。

Wang, Wang 和 Hu 的方法 该方法的基本结构包括两个模糊逻辑控制器：一个控制杂交率，一个控制变异率[655]。杂交率更新的启发式原则考虑了种群平均适应值的变化，记做 $\Delta f(t)$。如果满足下述 3 个条件：

1. 平均适应值的变化很小，即 $|\Delta f(t)|<\varepsilon$，其中 ε 是一个给定的小实数。
2. 平均适应值的变化大于 0，即 $\Delta f(t)>0$。
3. 平均适应值的变化在连续几代遗传中保持同号。

则杂交率增加；否则杂交率降低。如果平均适应值的变化接近 0，杂交率应迅速增加。模糊逻辑控制器的输入变量是 t 时刻平均适应值的变化 $\Delta f(t)$ 和 $t-1$ 时刻的 $\Delta f(t-1)$。可以利用这些参数计算 $\Delta^2 f(t)(\Delta^2 f(t)=\Delta f(t)-\Delta f(t-1))$。控制器的输出是杂交率的变化值 $\Delta c(t)$（参见表 1.1）。

表 1.1 杂交模糊决策表

$\Delta c(t)$		$\Delta f(t-1)$								
		NR	NL	NM	NS	ZE	PS	PM	PL	PR
$\Delta f(t)$	NR	NR	NL	NL	NM	NM	NS	NS	ZE	ZE
	NL	NL	NL	NM	NM	NS	NS	ZE	ZE	PS
	NM	NL	NM	NM	NS	NS	ZE	ZE	PS	PS
	NS	NM	NM	NS	NS	ZE	ZE	PS	PS	PM
	ZE	NM	NS	NS	ZE	ZE	PS	PS	PM	PM
	PS	NS	NS	ZE	ZE	PS	PS	PM	PM	PL
	PM	NS	ZE	ZE	PS	PS	PM	PM	PL	PL
	PL	ZE	ZE	PS	PS	PM	PM	PL	PL	PR
	PR	ZE	PS	PS	PM	PM	PL	PL	PR	PR

模糊输入和输出变量的隶属度函数如图 1.11 所示。术语的含义为：NR 代表负值非常大，NL 代表负值大，NM 代表负值中等，NS 代表负值小，ZE 代表零，PS 代表正值小，PM 代表正值中等，PL 代表正值大，PR 代表正值非常大。

表 1.2 是模糊逻辑控制器动作的检索表。在该表中，z 是不大于 $\alpha x+(1-\alpha)y$ 的最小整数，其中 α 为由适应值变化而变化的适应性系数。模糊逻辑控制器的输出由下式决定：

$$\Delta c(t) = z[x][y]\times 0.02\times\beta \tag{1.1}$$

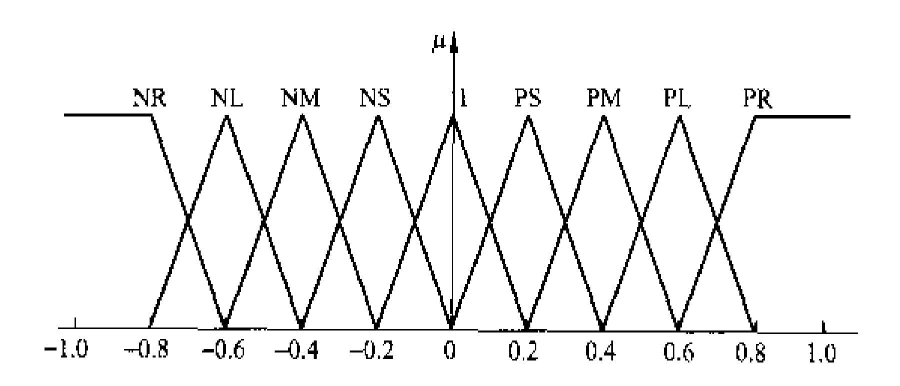


图1.11　$\Delta f(t)$、$\Delta f(t-1)$和$\Delta c(t)$的隶属度函数

其中β为另一个适应性系数。当整个种群适应值的变化小于0.02时，其数值小于1.0。因此杂交率由下式计算：

$$p_c(t) = p_c(t-1) + \Delta c(t) \tag{1.2}$$

表1.2　杂交控制动作的模糊隶属度值

z		x								
		−4	−3	−2	−1	0	1	2	3	4
y	−4	−4	−3	−3	−2	−2	−1	−1	0	0
	−3	−3	−3	−2	−2	−1	−1	0	0	1
	−2	−3	−2	−2	−1	0	0	1	1	2
	−1	−2	−2	−1	−1	2	1	1	2	2
	0	−2	−1	−1	0	1	1	2	2	3
	1	−1	−1	0	0	1	1	2	2	3
	2	−1	0	0	1	1	2	2	3	3
	3	0	0	1	1	2	2	3	3	4
	4	0	1	1	2	2	3	3	4	4

类似地，变异率的调整可由下式决定：

$$\Delta m(t) = z[x][y] \times 0.002 \times \beta \tag{1.3}$$

$$p_m(t) = p_m(t-1) + \Delta m(t) \tag{1.4}$$

Lee和Takagi的方法　在Lee和Takagi提出的动态参数遗传算法中[402]，模糊逻辑控制器的输入是任何遗传算法的性能判据或当前选取的参数值，输出则是遗传算法的任意参数值。针对无法获得专门知识的情况，作者提出了自动模糊设计方法来获取数据库和规则库。表现型的种群多样性度量用于监测遗传算法的性能[294]。设$\bar{f}$表示平均适应值，f_{best}表示最优适应值，f_{worst}表示最劣适应值。Lee和Takagi给出了两种表现型种群多样性度量方式：

$$\mathrm{PDM}_1 = \frac{\bar{f}}{f_{best}} \tag{1.5}$$

$$\mathrm{PDM}_2 = \frac{f_{worst}}{\bar{f}} \tag{1.6}$$

PDM_1 和 PDM_2 均属于区间[0,1]。如果它们趋于 1,则算法趋于收敛,如果它们趋于 0,则种群表现出高度的多样性。

用于调整策略性参数的启发式原则为:

1. 如果 PDM_1 大,则种群规模应增加。
2. 如果 PDM_2 小,则种群规模应减小。
3. 如果变异率小且种群规模小,则种群规模应增加。

Xu 和 Vukovich 的方法 在 Xu 和 Vukovich 提出的模糊进化算法中[673],模糊逻辑控制器的输入变量是两个参数:遗传代数和种群规模。输出为策略性参数:p_c 和 p_m。其策略性参数的调整规则如表 1.3 所示。

表 1.3 杂交和变异的模糊决策表

遗传代数	种群规模		
	小	中	大
短	大	中	小
中	中	小	很小
长	小	很小	很小

Herrera,Viedma,Lozano 和 Verdegay 的方法 他们提出的方法采用了分散统计度量作为输入变量[295]。设 L 表示染色体长度,N 表示种群规模,S_{ij} 表示染色体 i 在位置 j 上的基因,定义下列统计值:

$$\bar{S}_i = \frac{\sum_{j=1}^{L} S_{ij}}{L} \tag{1.7}$$

$$\bar{S} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{L} S_{ij}}{LN} \tag{1.8}$$

$$\bar{S}_j = \frac{\sum_{i=1}^{N} S_{ij}}{N} \tag{1.9}$$

然后定义染色体平均偏差 AVC 如下式:

$$\mathrm{AVC} = \frac{\sum_{i=1}^{N} (\bar{S}_i - \bar{S})^2}{N} \tag{1.10}$$

定义等位基因平均偏差 AVA 如下式:

$$\mathrm{AVA} = \frac{\sum_{j=1}^{L}\sum_{i=1}^{N} (\bar{S}_i - \bar{S}_j)^2}{N} \tag{1.11}$$

模糊规则给定如下：

1. 如果 AVC 低，则 p_c 应少量增加。
2. 如果 AVC 高，则 p_c 应少量降低。
3. 如果 AVA 低，则 p_m 应少量增加。
4. 如果 AVA 高，则 p_m 应少量降低。

Herrera 和 Lozano 的方法 Herrera 和 Lozano 提出了基于模糊逻辑控制器的适应性实数编码遗传算法，称作 ARGAF[294]。ARGAF 的主要特点为：

1. 该算法采用了两种不同的杂交算子：一种具有深度搜索特性，另一种具有广度搜索特性。
2. 该算法采用了线件排序选择机制[35]。
3. 采用模糊逻辑控制器对 p_e 和 η_{min} 这两个参数进行调整。

参数 p_e 定义为应用深度和广度搜索杂交算子的频率。参数 η_{min} 确定选择压力。如果 η_{min} 低，则获得较大的选择压力；反之，如果 η_{min} 高，选择压力小。

采用两种多样性度量作为输入。一种是基因型多样性度量 ED(Euclidean 距离)，另一种是表现型多样性度量 PDM_1。第一种度量代表了遗传物质在种群中的数量，而第二种度量则代表了多样性的质量。

设 P 表示种群集合，$\boldsymbol{v}_i$ 表示第 i 个个体，$\boldsymbol{v}^*$ 表示种群中的最优个体，$d(\boldsymbol{v}_i,\boldsymbol{v}_j)$ 表示两个个体间的 Euclidean 距离。定义

$$\bar{d}=\frac{1}{N}\sum_{i=1}^{N}d(\boldsymbol{v}^*,\boldsymbol{v}_i) \tag{1.12}$$

$$d_{max}=\max\{d(\boldsymbol{v}^*,\boldsymbol{v}_i)\mid \boldsymbol{v}_i\in P\} \tag{1.13}$$

$$d_{min}=\min\{d(\boldsymbol{v}^*,\boldsymbol{v}_i)\mid \boldsymbol{v}_i\in P\} \tag{1.14}$$

ED 定义为

$$\mathrm{ED}=\frac{\bar{d}-d_{min}}{d_{max}-d_{min}} \tag{1.15}$$

ED 的范围是[0,1]。如果 ED 小，种群中大多数个体集中在最优个体周围，即算法收敛。如果 ED 大，大多数个体并没有向当前最优个体收敛。

描述输入 ED 和 PDM_1 与输出 p_e 关系的规则如表 1.4 所示。描述输入 ED 和 PDM_1 与输出 η_{min} 关系的规则如表 1.5 所示。

表 1.4 参数 p_e 的模糊决策表

ED	PDM_1		
	低	中	高
低	小	小	中
中	大	大	中
高	大	大	中

表 1.5 参数 η_{min}的模糊决策表

ED	PDM₁		
	小	中	大
低	小	中	大
中	小	大	大
高	小	小	大

1.3 遗传优化

优化处理的是具有多个变量且通常需要服从等式和(或)不等式约束的最小化或最大化函数问题。在运筹学、管理科学和工程设计中,优化问题处于非常重要的地位。许多工业工程的设计问题非常复杂而困难,以至于难以采用传统优化方法进行求解。近年来,由于遗传算法作为新型优化方法具有良好的潜质,该算法受到普遍的关注。遗传算法具有简单、易操作、需求低、并行和全局性等特点,它已经在非常广泛的领域中取得了成功应用[147,219,456]。本节给出遗传优化方法的简单介绍,包括其优化的主要领域:全局优化、约束优化、组合优化、多目标优化等。

1.3.1 全局优化

全局优化(global optimizations)采用在感兴趣的区域内可以区分全局最优解和众多局部最优解的方法。全局优化问题通常具有无约束优化的形式,即问题除了一个最小化或最大化函数之外没有其他限制[45]。一般来说,无约束优化问题可以在数学上描述为[428]:

$$\begin{aligned} &\min \quad f(\boldsymbol{x}) \\ &\text{s.t.} \quad \boldsymbol{x} \in \Omega \end{aligned}$$

其中 f 是实值函数,可行集合 Ω 是$\mathbb{R}^n$的子集。可行集 $\Omega=\mathbb{R}^n$代表了完全无约束的情况。在许多应用问题中,我们仅仅需要考虑 Ω 是$\mathbb{R}^n$特定的子集的情况。如果存在 $\varepsilon>0$,对于所有满足属于 Ω 且与 $\boldsymbol{x}^*$ 的距离小于 ε 的 $\boldsymbol{x}$,都有 $f(\boldsymbol{x})\geqslant f(\boldsymbol{x}^*)$,则点 $\boldsymbol{x}^*$ 称作 f 在 Ω 上的局部最优解。如果对于所有 $\boldsymbol{x}\in\Omega$ 都有 $f(\boldsymbol{x})\geqslant f(\boldsymbol{x}^*)$,则点 $\boldsymbol{x}^*$ 称作 f 在 Ω 上的全局最优解。尽管大多数实际优化问题有需要满足的边界约束,对于无约束优化问题方法的研究为进一步研究打下了基础。

传统全局优化方法可以粗略地分为两类:(1)确定性方法,(2)随机性方法[623]。遗传算法在那些对于传统爬山法和基于导数的方法来说性质恶劣、不可微、不连续的函数优化方面取得了很大的成功。这类问题的例子包括多峰、不可微和不连续问题。自 20 世纪 70 年代早期遗传算法出现以来,全局优化就是其主要应用目标之一,同时也为开发有效的全局优化问题算法提供了许多帮助。

通常将遗传算法应用于全局优化问题的方式是将每个决策变量采用二进制或 Gray 方式进行编码。对于有 n 个变量的问题，$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$，带有 n 段字符串的染色体如下：

$$\underbrace{1001\cdots01}_{x_1}\ \underbrace{1001\cdots01}_{x_2}\ \cdots\ \underbrace{1001\cdots01}_{x_n}$$

因此，如果每个变量 x_i 编码为 l_i 个位，完整的染色体就具有长度 $\sum_{i=1}^{n} l_i$ 。在遗传算法中，采用二进制表示解的强烈偏好通常来自于遗传算法的模式理论(schema theory)。该理论试图用遗传算法的期望模式采样行为来分析算法的性能。然而，对于连续变量二进制编码的实际经验和理论分析明确表明：二进制表示具有严重缺陷。它通常会在目标函数中引入附加的多峰性，从而使编码后的目标函数比原始问题更加复杂[33]。因此，解决函数优化问题的主要方向是采用实数表示，有时称作实数编码遗传算法(real-coded genetic algorithms) [147,180,647]。这个方向开辟出一个非常活跃的研究领域。

实数表示中，每个染色体由实数向量组成。对于有 n 个变量的问题，对应的实数向量就是 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$。在过去的 20 年里，针对实数编码提出了若干遗传算子。这些算子可以粗略地分为下述 4 类：

1. 传统算子
2. 算术算子
3. 基于方向的算子
4. 随机算子

传统算子(conventional operators)仅将二进制表示的算子延伸到实数编码。算术算子从凸集理论中向量的线性组合概念得来。基于方向的算子则是将近似梯度(次梯度)方向引入遗传算子。随机算子利用某种分布(通常是 Guassian 分布)的随机数来改变父代从而获得子代。下面给出了一些经常使用的遗传算子的简介。

算术杂交 (arithmetical crossover)　这种算子的基本概念由凸集理论得来[44]。一般地，两个向量 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 的加权平均可以计算如下：

$$\lambda_1\boldsymbol{x}_1+\lambda_2\boldsymbol{x}_2 \tag{1.16}$$

如果乘子的限制为

$$\lambda_1+\lambda_2=1\qquad \lambda_1>0,\quad \lambda_2>0 \tag{1.17}$$

则加权形式(1.16)就称作凸组合。如果不满足乘子非负的约束，组合就称作仿射组合(affine combination)。如果乘子在实数空间 $\mathbb{R}$ 中，则组合就称作线性组合(linear combination)。

类似地，算术杂交定义为两个向量(染色体)的组合如下：

$$\boldsymbol{x}_1'=\lambda_1\boldsymbol{x}_1+\lambda_2\boldsymbol{x}_2 \tag{1.18}$$

$$\boldsymbol{x}_2'=\lambda_1\boldsymbol{x}_2+\lambda_2\boldsymbol{x}_1 \tag{1.19}$$

根据乘子的限制，出现了 3 种类型的杂交：凸杂交(convex crossover)、仿射杂交(affine crossover)和线性杂交(linear crossover)。下面从几何角度对算术杂交进行解释。对于父代 x_1 和 x_2，所有凸组合的集合构成凸锥(convex hull)。类似地，定义仿射锥(affine hull)为所有仿射组合的集合，线性锥(linear hull)为所有线性组合的集合。图 1.12 用二维空间中的简单情况表示出这些概念。凸杂交产生的后代位于实线上，仿射杂交产生的后代位于虚线上，线性杂交产生的后代位于整个解空间中。

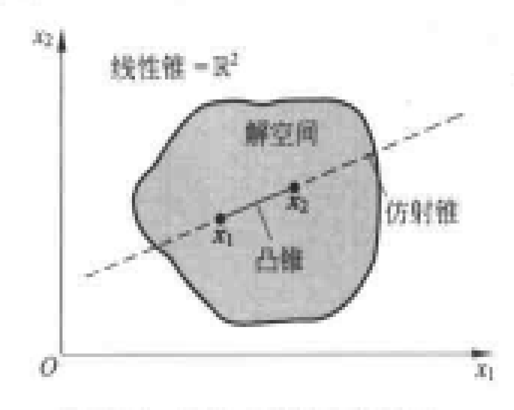


图 1.12 凸锥、仿射锥和线性锥

特别地，当 $\lambda_1=\lambda_2=0.5$ 时，Davis 称其为平均杂交(average crossover)[147]，Schwefel 称其为中间杂交(intermediate crossover)[566]。Wright 将乘子限制为 $\lambda_1=1.5,\lambda_2=-0.5$ 的形式，这是仿射杂交的一个特例[667]。作为另外一种仿射杂交，Mühlenbein 和 Schlierkamp-Voosen 提出了*扩展中间杂交*(extended intermediate crossover)。该方法把一个乘子作为区间$[-d, 1+d]$上的随机数。Cheng 和 Gen 将乘子限制为：

$$\lambda_1+\lambda_2\leqslant 2 \qquad \lambda_1>0,\quad \lambda_2>0$$

这是线性杂交的一个特例[118]。

混合杂交(blend crossover) 从本质上讲，混合杂交在由父代点构成的超三角形内随机创建后代[180]。考虑一维情况，即问题仅有一个变量。假设第一个父代的值为 p_1，第二个父代的值为 p_2，而且 $p_2>p_1$。设 $I=|p_1-p_2|$，且 $0<\alpha,\beta<1$，则后代在下式区间中随机选取一个点产生：

$$[p_1-\alpha I, p_2+\beta I] \tag{1.20}$$

上式通常被指定为 BXL$-\alpha-\beta$。图 1.13 解释了混合杂交。如果问题有两个变量，新的后代将位于有两个区间构成的三角中，如果问题有 n 个变量，后代将是由 n 的区间定义的超三角中的一个点。为了与 Eshelman，Mathias 和 Schaffer 提出的方向 BLX(directional-BLX)[179]相区别，这种混合杂交还称作盒子 BLX(box-BLX)。盒子 BLX 在由父代定义的超三角中均匀采样，而方向 BLX 则倾向于在与父代定义的对角线平行的区间上采样。这种处理方式基于下述假设。如果种群沿着问题空间的对角线分布，则沿着父代定义的对角线采样比从父代定义的盒子中采样更有意义。方向 BLX 可以表示为 BLX$-\alpha-\beta-\gamma$。描述方向 BLX 最简单的方法是将其考虑为盒子 BLX 的一种特例，区别在于与两个个体

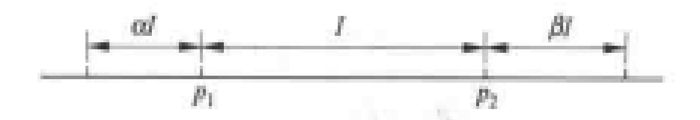


图 1.13 一维情况下的混合杂交

的对角线相对的角被截去了。参数 γ 代表盒子被截去的数量。图 1.14 说明了盒子 BLX 和方向 BLX。在 BXL—0.0—0.0—1.0 的情况中，采样区域被限制在由两个父代指定的盒子中，而在 BXL—0.0—0.0—0.5 的情况中，采样区域截去了与对角线相对的两个角。

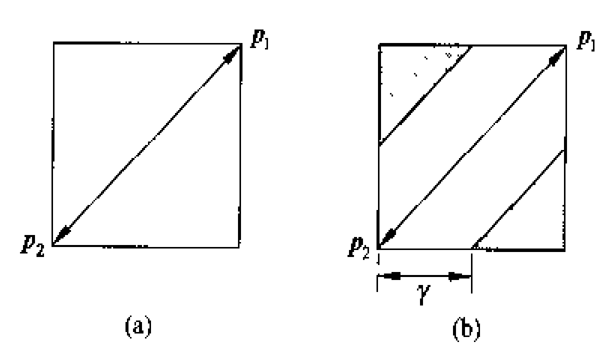


图 1.14　二维情况下的盒子 BLX 和方向 BLX

单峰正态分布杂交(unimodal normal distribution crossover)　Ono 和 Kobayshi 提出的 UNDX 杂交的理论基础与 Eshleman 的方向 BLX 类似。该杂交倾向于在由父代定义的对角线区域中进行采样。UNDX 在由 3 个父代定义的正态分布区域中产生两个后代。图 1.15表示了二维情况的 UNDX。在由两个父代 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 定义的方向中，正态分布的标准差与父代 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 的距离成比例。在与该方向正交的方向中，正态分布的标准差与父代 $\boldsymbol{p}_3$ 和第一个方向的距离成比例。为了减少第三个父代的影响，该距离除以$\sqrt{n}$。

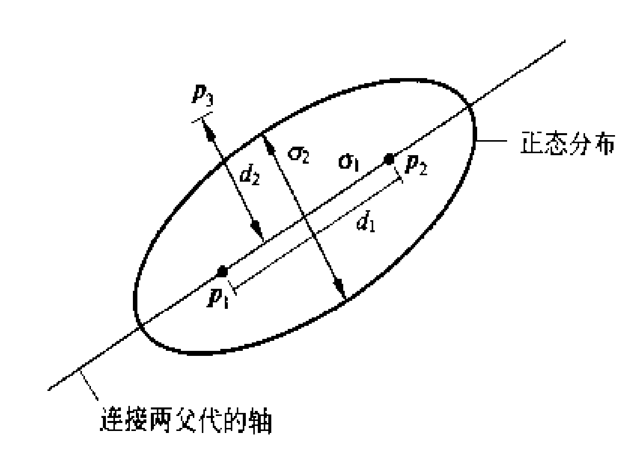


图 1.15　单峰正态分布杂交

设 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 表示父代向量，$\boldsymbol{c}_1$ 和 $\boldsymbol{c}_2$ 表示子代向量，n 表示变量数，d_1 表示父代 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 之间的距离，d_2 表示父代 $\boldsymbol{p}_3$ 与 $\boldsymbol{p}_1$ 和 $\boldsymbol{p}_2$ 之间连线的距离，z_1 表示服从正态分布 $N(0,\sigma_1^2)$ 的随机数，z_k 表示服从正态分布 $N(0,\sigma_k^2)(k=2,3,\cdots,n)$ 的随机数，α 和 β 是常数。子代由下式产生：

$$\boldsymbol{c}_1=\boldsymbol{m}+z_1\boldsymbol{e}_1+\sum_{k=2}^{n}z_k\boldsymbol{e}_k \tag{1.21}$$

$$\boldsymbol{c}_2=\boldsymbol{m}-z_1\boldsymbol{e}_1-\sum_{k=2}^{n}z_k\boldsymbol{e}_k \tag{1.22}$$

$$\boldsymbol{m}=\frac{\boldsymbol{p}_1+\boldsymbol{p}_2}{2} \tag{1.23}$$

$$z_k \sim N(0,\sigma_k^2),\quad k=1,2,\cdots,n \tag{1.24}$$

$$\sigma_1=\alpha d_1,\quad \sigma_2=\frac{\beta d_2}{\sqrt{n}} \tag{1.25}$$

$$\boldsymbol{e}_1=\frac{\boldsymbol{p}_2-\boldsymbol{p}_1}{|\boldsymbol{p}_2-\boldsymbol{p}_1|} \tag{1.26}$$

$$\boldsymbol{e}_i \perp \boldsymbol{e}_j,\quad i,j-1,2,\cdots,n,\quad i\neq j \tag{1.27}$$

UNDX与坐标系统相对独立，原因是它是基于父代的连线而不是坐标系统的轴来生产子代的。

边界算子(boundary operators) 该方法由 Schoenauer 和 Michalewicz 提出[565]。这种算子基于如下的结论：许多优化问题的全局最优解通常存在于可行区域的边界上。因此，对于许多约束优化问题来说，仅搜索由一系列约束定义的解空间的边界可能效果更好。球杂交(sphere crossover)就是这种方法的一个例子。该方法采用下式由两个父代$(x_1,x_2,\cdots,x_n)$和$(y_1,y_2,\cdots,y_n)$生成一个子代$(z_1,z_2,\cdots,z_n)$：

$$z_i=\sqrt{\alpha x_i^2+(1-\alpha)y_i^2}\quad i=1,2,\cdots,n \tag{1.28}$$

其中α是从$[1,\eta]$区间内选择的随机数。

基于方向的杂交(direction-based crossover) 这种方法采用目标函数的值来确定遗传搜索的方向[456]。该算子采用下式由两个父代$\boldsymbol{x}_1$和$\boldsymbol{x}_2$产生一个子代$\boldsymbol{x}'$：

$$\boldsymbol{x}'=r(\boldsymbol{x}_2-\boldsymbol{x}_1)+\boldsymbol{x}_2 \tag{1.29}$$

其中r是0和1之间的随机数。该算子同时假设父代$\boldsymbol{x}_2$不差于$\boldsymbol{x}_1$，即对最大化问题来说，$f(\boldsymbol{x}_2)\geqslant f(\boldsymbol{x}_1)$，对于最小化问题来说，$f(\boldsymbol{x}_2)\leqslant f(\boldsymbol{x}_1)$。

非均匀变异(nonuniform mutation) 这种方法由 Janilow 和 Michalewicz 提出[455]。它的设计目的是为了得到较高的精确度而具有微调能力。对于给定的父代$\boldsymbol{x}$，如果它的元素被x_k选中进行变异，结果的后代是$\boldsymbol{x}'=(x_1,\cdots,x_k',\cdots,x_n)$，其中$x_k'$从下面两种可能的方案中随机选择：

$$x_k'=x_k+\Delta(t,x_k^U-x_k) \tag{1.30}$$

$$x_k'=x_k-\Delta(t,x_k-x_k^L) \tag{1.31}$$

函数$\Delta(t,y)$返回范围$[0,y]$中的一个值，当遗传代数t增加时，该值越来越趋向于0。该特性导致算子在早期(t很小)均匀地搜索解空间，而到了晚期则在很小的区域进行搜索。函数$\Delta(t,y)$的形式如下：

$$\Delta(t,y)=yr\left(1-\frac{t}{T}\right)^b \tag{1.32}$$

其中r是$[0,1]$区间上的随机数，T是最大遗传代数，b是确定非均匀程度的参数。该算子有可能产生不可行的后代。如果这种情况发生，可以减小随机数r的数值。

有向变异(directional mutation) 这种方法由 Gen，Liu 和 Ida 提出[237,238]。设$\boldsymbol{d}$表

示近似的梯度方向。变异后的后代可以表示为

$$x' = x + rd$$

其中 r 是随机非负实数,它的选取要保证后代是一个可行解。近似梯度的第 i 个元素由下式计算:

$$\frac{f(x_1,\cdots,x_i+\Delta x_i,\cdots,x_n)-f(x_1,\cdots,x_i,\cdots,x_n)}{\Delta x_i} \tag{1.33}$$

其中 Δx_i 是一个小实数。

为了避免染色体陷落人解空间中的角落(corner),也可以将该方向随机选取。如果染色体接近边界,由某些判据给出的变异方向可能指向闭边界,因此发生陷落现象。这种情况发生时就采用随机方向来替代给定的方向。

高斯变异(Gaussian mutation) 这种方法起源于进化策略[31,32]。一般在进化策略中的一个个体包含两个元素($\boldsymbol{x},\boldsymbol{\sigma}$),其中第一个向量 $\boldsymbol{x}$ 表示搜索空间中的一个点,而第二个向量 $\boldsymbol{\sigma}$ 表示标准差。后代($\boldsymbol{x}',\boldsymbol{\sigma}'$)由下式产生:

$$\boldsymbol{\sigma}' = \boldsymbol{\sigma}\mathrm{e}^{N(\mathbf{0},\Delta\boldsymbol{\sigma})} \tag{1.34}$$

$$\boldsymbol{x}' = \boldsymbol{x} + N(\mathbf{0},\Delta\boldsymbol{\sigma}') \tag{1.35}$$

其中 $N(\mathbf{0},\Delta\boldsymbol{\sigma}')$是均值为 $\mathbf{0}$,标准差为 $\boldsymbol{\sigma}$ 的独立高斯随机数向量。

1.3.2 约束优化

非线性规划(或约束优化(constrained optimizations))处理的是有等式和(或)不等式约束的优化目标函数的问题。由于许多实际问题无法建模为线性规划问题,因此非线性规划成为在几乎所有工程、运筹学和数学领域中非常重要的工具。非线性规划的一般形式如下:

$$\max \quad f(\boldsymbol{x}) \tag{1.36}$$

$$\text{s.t.} \quad g_i(\boldsymbol{x}) \leqslant 0, \quad i - 1,2,\cdots,m_1 \tag{1.37}$$

$$h_i(\boldsymbol{x}) = 0, \quad i = m_1+1,\cdots,m(=m_1+m_2) \tag{1.38}$$

$$\boldsymbol{x} \in X \tag{1.39}$$

其中,$f,g_1,g_2,\cdots,g_{m_1},h_{m_1+1},h_{m_1+2},\cdots,h_m$ 是在$\mathbb{R}^n$上定义的实值函数,X 是$\mathbb{R}^n$的子集,$\boldsymbol{x}$ 是 n 维实向量,其元素为 $x_1,x_2,\cdots,x_n$。上述问题的解必须为既能满足约束同时又最小化函数 f 的变量 $x_1,x_2,\cdots,x_n$。函数 f 通常称作目标函数(objective function)或判据函数(criterion function)。约束 $g_i(\boldsymbol{x})\leqslant 0$ 称作不等式约束(inequality constraint),约束 $h_i(\boldsymbol{x})=0$称作等式约束(equality constraint)。集合 X 通常可能包括变量的下界和上界,称作域约束(domain constraint)。满足所有约束的向量 $\boldsymbol{x}\in X$ 称作问题的可行解(feasible solution)。这种解的集合构成可行区域(feasible region)。非线性规划问题就是要找到一个可行点 $\bar{\boldsymbol{x}}$,并对每一个可行点 $\boldsymbol{x}$,都有 $f(\boldsymbol{x})\leqslant f(\bar{\boldsymbol{x}})$。这种点称作最优解

(optimal solution)。与线性规划不同，非线性规划的传统解法非常复杂，而且效率不高。在过去的几年里，将遗传算法用于解决非线性规划问题成为一种潮流[537]。本节将略述如何用遗传算法解决非线性规划问题。

用于操作染色体的遗传算子通常会产生非线性规划的不可行后代，因此将遗传算法应用于非线性规划的主要问题就是如何处理约束。最近提出了若干在遗传算法中处理约束的方法[452,494]。现存的方法可以粗略地分类如下：拒绝方法(rejecting methods)、修补方法(repairing methods)和罚方法(penalty methods)。拒绝方法抛弃进化过程中产生的所有不可行解。这是处理问题最简单但也是效率最低的方法。修补方法包括获得不可行解和通过修补过程产生可行解两部分。对于许多组合优化问题，创建修补过程相对较容易。

罚方法可能是遗传算法用于约束优化问题最常用的方法。从本质上讲，这种方法通过对不可行解的惩罚来将约束问题转换为无约束问题。任何对约束的违反都要在目标函数中添加惩罚项。

罚方法的基本思想从传统优化中借鉴而来。很自然会问：我们在传统优化中采用罚方法和在遗传算法中采用罚方法有什么区别？传统优化中，罚方法用于产生一系列不可行点，其极限就是原始问题的最优解。考虑的焦点集中在如何选择合适的罚值，从而加速收敛并且防止早熟终止。遗传算法中，罚方法用于在每一代中维持一定数量的不可行解，从而使遗传搜索从可行区域和不可行区域两个方向搜索最优解。通常并不拒绝每代中的不可行解，原因在于其中一些个体可能提供关于最优解的更有用的信息。我们的关注点在于如何确定惩罚项，从而在信息保留(保持一些不可行解)与选择压力(拒绝一些不可行解)之间维持平衡，并要注意避免惩罚不够或者过度惩罚。

一般来说，解空间包含两部分：可行区域和不可行区域。我们对于这些子空间没有任何前提假设。特别地，它们不必如图 1.16 一样是凸的或连通的。处理不可行染色体远不是微不足道的工作。从图中可以看出，不可行解 b 远比不可行解 d 和可行解 c 离最优解 a 要近。尽管 b 比 d 离可行区域的距离大，但我们希望对 b 的惩罚比对 d 的惩罚要小。我们也意识到尽管 b 是不可行解，但它带有最优解的信息要比 c 多。然而，我们对最优解没有任何先验知识。因此一般来说判断哪个解是最优的非常困难。罚方法的主要问题就是如何设计罚函数 $p(x)$，使得它能够有效地将遗传搜索引导到解空间中有希望的区域中去。不可行染色体和搜索空间的可行部分之间的关系在惩罚不可行染色体时起着重要作用。惩罚值根据某种度量反映了不可行的程度。关于设计罚函数(penalty function)没有一般的指导性原则，构造一个有效的罚函数依赖于给定的问题。

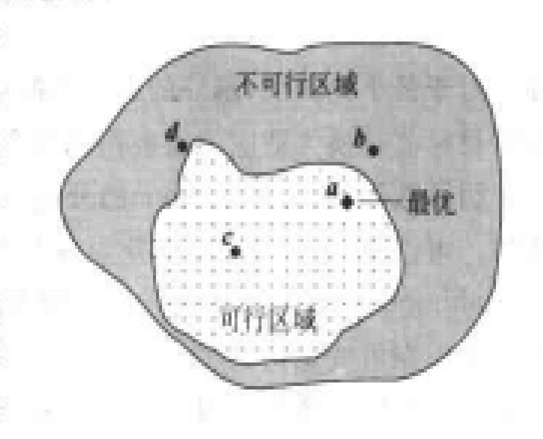


图 1.16 解空间：可行区域和不可行区域

评价带有惩罚项的函数　罚方法通过将惩罚不可行解将约束问题转换为无约束问题。一般来说，有两种可行的建立带有惩罚项评价函数的方法。一种是采用加的形式，即

$$\text{eval}(\boldsymbol{x}) = f(\boldsymbol{x}) + p(\boldsymbol{x}) \tag{1.40}$$

其中 $\boldsymbol{x}$ 表示一条染色体，$f(\boldsymbol{x})$是问题的目标函数，$p(\boldsymbol{x})$是惩罚项。对于最大化问题，通常要求：

$$\begin{aligned} p(\boldsymbol{x}) &= 0, \quad \text{如果 } \boldsymbol{x} \text{ 可行} \\ p(\boldsymbol{x}) &< 0, \quad \text{否则} \end{aligned} \tag{1.41}$$

设$|p(\boldsymbol{x})|_{\max}$和$|f(\boldsymbol{x})|_{\min}$分别表示当前种群中最大的$|p(\boldsymbol{x})|$和最小的$|f(\boldsymbol{x})|$。同时要求

$$|p(\boldsymbol{x})|_{\max} \leqslant |f(\boldsymbol{x})|_{\min} \tag{1.42}$$

从而避免产生负的适应值。对于最小化问题，通常要求：

$$\begin{aligned} p(\boldsymbol{x}) &= 0, \quad \text{如果 } \boldsymbol{x} \text{ 可行} \\ p(\boldsymbol{x}) &> 0, \quad \text{否则} \end{aligned} \tag{1.43}$$

第二种方式是采用乘的形式，即

$$\text{eval}(\boldsymbol{x}) = f(\boldsymbol{x})p(\boldsymbol{x}) \tag{1.44}$$

对于这种情况，如果是最大化问题，则要求

$$\begin{aligned} p(\boldsymbol{x}) &= 1, \quad \text{如果 } \boldsymbol{x} \text{ 可行} \\ 0 \leqslant p(\boldsymbol{x}) &< 1, \quad \text{否则} \end{aligned} \tag{1.45}$$

如果是最小化问题，则要求

$$\begin{aligned} p(\boldsymbol{x}) &= 1, \quad \text{如果 } \boldsymbol{x} \text{ 可行} \\ p(\boldsymbol{x}) &> 1, \quad \text{否则} \end{aligned} \tag{1.46}$$

注意对于最小化问题，越好的染色体的 eval($\boldsymbol{x}$)函数值越小。对于某些选择方式而言，需要将目标值转换为适应值以确保优秀个体具有高的适应值。

罚函数(penalty function)的分类　在遗传算法领域中提出了若干种处理不可行解的方法。可以将这些方法大致分为两类：(1)*常数惩罚*(constant penalty)，(2)*可变惩罚*(variable penalty)。已知对于复杂问题来说常数惩罚效率不高。最近的研究工作都将注意力集中在可变惩罚上。

可变惩罚通常包含两个元素：(1)可变的惩罚率，(2)对于违背的约束的惩罚量。可变的惩罚率可以根据违背约束的程度和遗传算法的迭代数来调整。根据 Michalewicz 的讨论[453]，第一种方法随着违背程度的加深而增加惩罚压力(这种方法称作*静态惩罚*(static penalty))，第二种方法随着进化过程而增加惩罚压力(这种方法称作*动态惩罚*(dynamic penalty))。

惩罚通常是个体与可行域之间距离的函数，可由下述3种方式给出：

1. 作为单个不可行解绝对距离的函数。

2. 作为当前种群中所有不可行解相对距离的函数。

3. 作为适应性惩罚项的函数。

大多数方法采用第1种方式。对于约束很强的问题，从不可行到可行的惩罚项在每代中相对较强。这时由于需要在保持信息和保持对不可行解压力之间取得平衡，就可能采用第2种和第3种方式。

罚方法还可以进一步分为依赖于问题的(problem dependent)和不依赖于问题的(problem independent)。大多数罚方法属于依赖于问题的方法。

罚方法还可以由带有参数或不带参数来区分。大多数罚方法是需要参数的方法。带有参数的罚方法一般是依赖于问题的方法。

1.3.3 组合优化

组合优化(combinatorial optimization)的特点是可行解的数量有限。虽然从理论上讲这种有限问题可以通过简单枚举找到最优解，然而在实际情况中通常无法实现。特别当实际问题的规模很大，可行解的数量非常多时更是如此。解决这种困难问题的主要方法是采用启发式搜索。在过去的10年中，遗传算法受到组合优化界人士越来越多的关注，并且在研究和许多工业工程领域的实际应用中表现出很强的生命力。

组合优化包含具有不同特征和属性的一大类问题。虽然这些问题彼此不同，然而问题的本质可以归纳为下列类型之一：

- 确定与问题相关的某些项的排列
- 确定某些项的组合
- 确定某些项的排列和组合
- 任何带有约束的上述类型

举例来说，资源约束的项目调度问题(resource-constrained project scheduling problems)和车辆路径和调度问题(vehicle routing and scheduling problem)的本质都是确定满足约束的某些项的排列，集覆盖问题(set-covering problem)和群体问题(grouping problem)的本质都是确定某些项的组合，并行机器调度问题(parallel machine scheduling problem)的本质是确定满足约束的某些项的排列和组合。

组合优化问题常见的特点是如果排列和(或)组合确定了，就可以方便地通过依赖于问题的过程来确定解。因此一般采用遗传算法来处理这类问题的方法是：

1. 采用遗传算法来进化所考虑项的排列和(或)组合。

2. 采用启发式方法根据排列和(或)组合来构造一个解。

将遗传算法应用于组合优化问题既是富有挑战性的又是困难的。关键的问题是如何将问题的解编码为染色体。由于组合优化问题中带有复杂的约束，简单的二进制串根本不能工作，原因在于它必然会产生不可行解甚至非法解。大多数研究人员最初的努力都

是对问题提出新的有效编码方式，这就诞生了非常活跃的研究领域：基于次序的遗传算法(order-based genetic algorithms)[249]。第2章中我们将介绍如何用遗传算法解决某些经典组合优化问题，包括集覆盖问题，装箱问题(bin packing problem)，背包问题(knapsack problem)和最小生成树问题(minimum spanning tree problem)。

1.3.4 多目标优化

自20世纪60年代初以来多目标优化(multiobjective optimizations)问题就受到研究人员越来越多的关注。在多目标优化问题中，多个目标函数需要同时进行优化。由于目标之间的无法比较和矛盾等现象，导致不一定存在在所有目标上都是最优的解。某个解可能在一个目标上是最优的但在另一个上是最差的。因此，多目标问题通常存在一个解的集合，它们之间不能简单地进行比较好坏。对这种解来说，不可能使得在任何目标函数上的改进不损害至少一个其他目标函数，这种解称作非支配解(nondominated solutions)和Pareto最优解(Pareto optimal solutions)。

在过去的几年中，将遗传算法应用于多目标优化问题成为研究热点，这种算法通常称作进化多目标优化(evolutionary multiobjective optimization)或遗传多目标优化(genetic multiobjective optimization)。遗传算法的基本特点是多方向和全局搜索，带有潜在解的种群因此能够一代一代地维持下来。从种群到种群的方法对于搜索Pareto解来说是有益的。

关于用遗传算法来解决多目标优化问题中涌现出的一个特别问题是，如何根据多个目标来确定个体的适应值。过去10年广泛研究了适应值分配机制，并提出和测试了许多种方法。这个问题将在第3章详细讨论。

1.4 近期遗传算法的论文

在过去的10年中，有超过200篇对研究遗传算法或者将遗传算法应用于从生物到工程优化等不同领域的博士学位论文发表。不完整的统计数据表明了世界范围内大学中与遗传算法相关研究的发展趋势。遗传算法是从进化理论的原理中产生的，最早应用于函数优化问题。最近发现它在解决生物领域中许多问题时非常有用。比如说，遗传算法已经被应用于预报蛋白质的第三结构和解释神经元的电穿刺行为。关于遗传算法在生物领域应用的论文参阅表1.6。

组合优化问题是遗传算法研究界的主要研究对象。遗传算子在许多难以解决的问题上表现出很强的能力，并得到了很有价值的解。关于遗传算法在组合优化领域应用的论文参阅表1.7。

表 1.6 遗传算法在生物领域应用的论文

年 份	作 者	论 文 题 目
1993	Grand	The application of the genetic algorithm to protein tertiary structure prediction[283]
1994	Xiao	Computer assisted drug design: genetic algorithms and structures of molecular clusters of aromatic hydrocarbons and actinomycin D-deoxyguanosine[670]
1995	Besson	An evaluation of the genetic algorithm as a computational tool in protein NMR[60]
1996	Bangalore	Data analysis strategies for qualitative and quantitative determination of organic compounds by Fourier transform infrared spectroscopy[42]
1996	Eichler	On the development and interpretation of parameter manifolds for biophysically robust compartmental models of CA3 hippocampal neurons[173]
1996	Hase	Simulation on mutual adaptation of a body form and biped walking by nerve mustle skelation model and genetic algorithms[288]
1996	Lohn	Automated discovery of self-replicating structures in cellular space automata models[423]
1996	Parrill	Applications of artificial intelligence in drug design[504]
1997	Ho	Simulation of condensed-phase physical, chemical, and biological systems on massively parallel computers[301]
1997	Rocha	Evidence sets and contextual genetic algorithms: exploring uncertainty, context, and embodiment in cognitive and biological systems[538]
1998	Niesse	The structural optimization of atomic and molecular icro clusters using a genetic algorithm in real-valued space-fixed coordinates[484]
1998	Palazolo	Use of genetic algorithms in boirnded search for design of biological nitrification/denitrification waste treatment systems[497]

表 1.7 遗传算法在组合优化领域应用的论文

年 份	作 者	论 文 题 目
1994	Levine	Parallel genetic algorithm for the set partitioning problem[406]
1995	Lin	High-quality tour hybrid genetic schemes for TSP optimization problems[417]
1995	Tagami	Study on application method of genetic algorithms for combinatorial optimization problems[606]
1995	Nakamura	Study on protocol design and combinatorial optimization in resource assignment type problems on the basis of net theory and genetic algorithms[480]
1998	Ikonen	Genetic algorithm for a three-dimensional *non-convex* bin packing problem[317]
1999	Zhou	Study on constrained spanning tree problem with genetic algorithms[896]

遗传算法用于解决计算机科学领域里的许多问题，包括多计算机的并行和调度程序，分布式数据库设计等。关于遗传算法在计算机科学领域应用的论文参阅表1.8。

表1.8　遗传算法在计算机科学领域应用的论文

年　份	作　者	论　文　题　目
1994	Adar	Allocation and scheduling on multi-computers using genetic algorithms[5]
1994	Brown	Optimal rate concept acquisition using version spaces and genetic algorithms[78]
1995	Rho	Distributed database design: allocation of data and operations to nodes in distributed database systems [536]

控制工程系统的动态行为是困难而有挑战性的问题。最近遗传算法应用于控制设计：设计问题被描述为参数搜索问题，遗传算法用于寻找合适的控制参数和近似轨迹并由此产生良好的控制行为。关于遗传算法在工程控制领域应用的论文参阅表1.9。

表1.9　遗传算法在工程控制领域应用的论文

年　份	作　者	论　文　题　目
1994	Jeon	Genetic algorithm based non-linear modeling and its application to control and mechanical diagnostics[329]
1995	Hachino	Study on indentification of continuous time delay system and lower dimensionize of models by genetic algorithms[275]
1995	Memon	Optimization methods for real-time traffic control[449]
1995	Munemoto	Study on disperse control type dynamic load balance using genetic algorithm[473]
1995	Yang	Study on signature verification using a genetic algorithm method [678]
1996	Cheng	Control design and robustness measurement for biped locomotion[102]
1997	Abido	Intelligent techniques approach to power systems identification and control[1]
1997	Hajda	Genetic algorithms for control of wastewater conveyance systems[276]
1997	Marchelya	Enforcing pollution control laws: Stackelberg Markov game models for improving efficiency and effectiveness[437]
1997	Masters	Evolutionary design of controlled structures[443]

工程设计吸引了遗传算法大量的研究和应用。比如拓扑结构设计、网络设计、VLSI布局、动态系统设计与集成、非线性滤波器、设备定位、制造元设计、污染控制、涡轮刃片夹具运动学设计等。关于遗传算法在工程设计领域应用的论文参阅表1.10。

表 1.10 遗传算法在工程设计领域应用的论文

年 份	作 者	论 文 题 目
1992	Bowden	Genetic algorithm based machine learning applied to the dynamic routing of discrete parts[72]
1992	Jensen	Topological structural design using genetic algorithms[327]
1992	Ros	Learning Boolean functions with genetic algorithms: a PAC analysis[541]
1992	Yao	Parameter estimation for nonlinear systems[679]
1993	Lee	Three new algorithms for exact D-optimal design problems[394]
1994	Fluerent	Algorithmes génétiques hybrides pour l'optimisation combinatoire[192]
1994	Palmer	Approach to a problem in network design using genetic algorithms[499]
1994	Shahookar	Application of the genetic algorithm for computer-aided design of VLSI layout[573]
1995	Abuali	Using determinant and cycle basis schemes in geneti algorithms for graph and network applications[2]
1995	Crossley	Using genetic algorithms as an automated methodology for conceptual design of rotorcraft[138]
1995	Huang	Data-driven description of reservoir petrophysical properties[311]
1995	Nims	Contingency ranking for voltage stability using a genetic algorithm[485]
1995	Noga	Performance analysis and simulation of a class of numerical demodulation techniques for large deviation FM signals[486]
1995	Pheatt	Construction of D-optimal experimental designs using genetic algorithms[508]
1995	Sun	Evolving population based search algorithms through thermodynamic operation: dynamic system design and integration[600]
1995	Tay	Automated generation and analysis of dynamic system designs[617]
1995	Tomasz	Nonlinear adaptive filtering: the genetic algorithm approach[624]
1995	Yu	Crustal deformation due to a dipping fault in an elastic gravitational layer overlying a viscoelastic gravitational half space[687]
1996	Brown	Using the facility location problem to explore operator policies and constraint-handling methods for genetic algorithms[77]
1996	Chaer	Mixture-of-experts approach to adaptive estimation[90]
1996	Joines	Hybrid genetic search for manufacturing cell design[332]
1996	Larson	Vibration testing by design: excitation and sensor placement using genetic algorithms[391]
1996	Lee	Genetic algorithms in multidisciplinary design of low vibration rotors[400]
1996	Li	Genetic algorithms-based design of multiplierless 2-d state-spase digital filters[415]
1996	Liu	Incorporating optimal design of groundwater remediation systems into numerical prediction[422]

续表

年份	作者	论文题目
1996	Lu	Study of genetic algorithms on an application to industrial design[426]
1996	Mock	Intelligent information filtering via hybrid techniques: hill climbing, case-based reasoning, index patterns, and genetic algorithms[458]
1996	Monem	Performance evaluation and optimization of irrigation canal system using genetic algorithm[460]
1996	Patel	Genetic algorithms: a tool for quantitative structure activity relationship analyses[505]
1996	Starns	Optimal synthesis of a planar four-bar mechanism with prescribed timing using generalized reduced radient, simulated annealing and genetic algorithms[585]
1996	Streifel	Application of computational intelligence to electromechanical systems[591]
1997	Houck	Meta-heuristics for manufacturing problems[309]
1997	Lazio	Genetic algorithms, pulsar planets, and ionized interstellar micro-turbulence[392]
1997	Maifeld	Genetic based unit commitment algorithm[433]
1997	Mantawy	Unit commitment by artificial intelligence techniques[436]
1997	Marchelya	Enforcing pollution control laws: Stackelberg-Markov game models for improving efficiency and effectiveness[437]
1997	Tang	Genetic algorithms for optimal operation of soil aquifer treatment systems[615]
1998	Gold	The application of the genetic algorithm to the Kinematic design of turbine blade fixtures[247]
1998	Guan	Applications of genetic algorithms in groundwater quality management[271]
1998	Jallas	Improved model-based decision support by modeling cotton variability and using evolutionary algorithms[323]
1999	Sasaki	Studies on solution methods for fuzzy reliability design problem[560]

优化问题几乎在各个领域中都存在，特别在工程界成为关注的焦点。工程界提出的许多优化问题非常复杂，同时难以用传统优化方法求解。遗传算法作为复杂问题优化方法的潜质已经受到了普遍的关注，已经成功应用于工业工程领域。关于遗传算法在工程优化领域应用的论文参阅表 1.11。

表 1.11 遗传算法在工程优化领域应用的论文

年 份	作 者	论 文 题 目
1991	Annaiyappa	Critical analysis of genetic algorithms for global optimization[16]
1992	Lee	Tolerance optimization using genetic algorithm and approximated simulation[399]
1993	Yunker	The optimization of simulation models by genetic algorithms: a comparative study[689]
1994	Hart	Adaptive global optimization with local search[287]
1994	Nakanishi	Optimizction of a structure phase by homology theory and genetic algorithms[481]
1995	Areibi	Towards optimal circuit layout using advanced search techniques[19]
1995	DeChaine	Stochastic fuel management optimization using genetic algorithms and heuristic rules[152]
1995	Dozier	Constraint processing using adaptive microevolutionary/systematic hill-climbing[156]
1995	Golden	Adaptive approximation and optimization of transform functions[253]
1995	Maria	Genetic algorithm for multimodel continuous optimization problems[438]
1995	Nolte	Global optimization algorithms for seismic inversion[487]
1995	Pinon	An investigation of the applicability of genetic algorithms to spacecraft trajectory optimization[511]
1995	Tompkins	Optimization of qualitative variables in discrete event simulation models[625]
1995	Yokota	Study on solving system reliability optimization problem with interval data by genetic algorithms [683]
1996	Liu	Application of multiobjective genetic algorithms to control systems design[422]
1996	Michael	Form-finding analysis of vibrating towered shells of revolution as an inverse eigen problem and as a genetic algorithm optimization problem[451]
1996	Moon	High performance simulation-based optimization environment for large-scale systems[456]
1996	Murata	Genetic algorithms for multi-objective optimization[457]
1997	Canpolat	Optimization of seasonal irrigation scheduling by genetic algorithms[84]
1997	Filho	Optimizing hydrocarbon field development using a genetic algorithms based approach[188]
1998	Matthew	Applications and limitations of genetic algorithms for the optimization of multivariate calibration techniques[445]

自 20 世纪 70 年代早期 Holland 首次提出遗传算法以来，大量的工作集中在从理论上研究遗传算法为什么能工作以及如何工作。基础的研究工作包括编码和表示、变异和

重组、适应值取值范围和遗传操作、选择和收敛、并行化、欺骗问题、遗传多样性、参数适应性等方面。关于遗传算法基础理论研究的论文参阅表1.12。

表1.12 遗传算法基础理论研究的论文

年份	作者	论文题目
1992	Collins	Studies in artificial evolution[131]
1992	Shu	Impact of data structures on the performance of genetic-algorithm-based learning[575]
1993	Kommu	Enhanced genetic algorithms in constrained search spaces with emphasis in parallel environments[373]
1993	Rankin	Consideration for rapidly converging genetic algorithms designed for application to problems with Mexpensive evaluation functions[543]
1994	Corcoran	Techniques for reducing the disruption of superior building blocks in genetic algorithms[143]
1994	Giddens	Determination of invariant parameters and operators of the traditional genetic algorithm using an adaptive genetic algorithm generator[244]
1994	Gordon	Exploitable parallelism in genetic algorithms[258]
1995	Cavaretta	Cultural algorithms and real-valued function optimization[87]
1995	Cedeno	Multi-niche crowding genetic algorithm: analysis and applications[88]
1995	Chung	Analysis of the effects of different representation schemes on genetic algorithms[124]
1995	Jones	The Hereford Ranch algorithm: an improvement of genetic algorithms using selective breeding[340]
1995	Mahfoud	Niching methods for genetic algorithms[432]
1995	Meddin	Genetic algorithms: A Markov chain and detail balance approach[448]
1995	Takashi	Study of evolutional mechanism on cooperative problem solving: a hybrid genetic algorithm for multi-algorithm problem[610]
1995	Wu	Non-coding DNA and floating building blocks for the genetic algorithm[669]
1996	Crawford	Role of recombination in genetic algorithms for fixed-length subset problems[137]
1996	Eberlein	GA heuristic generically has hyperbolic fixed points[169]
1996	Hightower	Computational aspects of antibody gene families[297]
1996	Kargupta	Search, polynomial complexity, and the fast messy genetic algorithm[346]
1996	Merkle	Analysis of linkage-friendly genetic algorithms[450]
1996	Picardo	Framework for the analysis of evolutionary algorithms[509]
1996	Sato	TA proposal and analysis on the alternation of generetion model of genetic algorithms[561]
1996	Stumpf	Studies on enhanced operator-oriented genetic algorithms[592]
1996	Wang	Matrix genome encoding for genetic algorithms[654]
1996	Wong	Performance analysis for genetic algorithms[666]
1996	Xie	Properties and characteristics of genetic algorithms as a problem-solving method[671]

续表

年 份	作 者	论 文 题 目
1997	Harik	Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms[286]
1997	Kaiser	Dynamic load distributions for adaptive computations MIMD machines using hybrid genetic algorithms[342]
1997	Potter	Design and analysis of a computational model of cooperative co-evolution[513]
1997	Rosin	Coevolutionary search among adversaries[542]
1998	Rasheed	GADO: a genetic algorithm for continuous design optimization[527]
1998	Sarma	Analysis of decentralized and spatially distributed genetic algorithms[559]
1998	Spears	Role of imitation and recombination in evolutionary algorithms[580]
1998	Voicu	Multiresolution aspects in genetic algorithms[643]

最近流行将模糊概念和遗传算法相结合形成模糊遗传算法(fuzzy genetic algorithms)。已经提出了基于模糊逻辑方法的两个应用：对遗传算法不同元素的建模和在不精确环境下求解问题。关于遗传算法和模糊逻辑的论文参阅表 1.13。

表 1.13 遗传算法和模糊逻辑的论文

年 份	作 者	论 文 题 目
1994	Lee	Automatic design and adaptation of fuzzy systems and genetic algorithms using soft computing techniques[401]
1994	Takagi	A Study on automatic structure method of fuzzy reasoning controller using neural networks and genetic algorithms[609]
1995	Ye	Fuzzy modeling of nonlinear systems and engenics-based genetic algorithm[681]
1996	Chbat	Direct learning neural and fuzzy control of two unknown dynamic systems using Powell optimization and genetic algorithms[99]
1996	Velthuizen	Feature extraction with genetic algorithms for fuzzy clustering[637]
1996	Xu	Genetic algorityms' seaching capability and their application for optimization of a model refernce fuzzy adaptive controller for linear systems with time delay[674]
1997	Egan	Validity-guided robust fuzzy clustering methods for characterizing clusters in noisy data[172]
1997	Inoue	Study on automatic generation method of fuzzy rules by genetic algorithms[318]
1997	Li	Oil tanker markets modeling, analysis and forecasting using neural networks, fuzzy logic and genetic algorithms[409]
1998	Li	Multiobjective optimum design of static and seismic resistant structures with genetic algorithms , fuzzy logic and game tbeory[407]

续表

年 份	作 者	论 文 题 目
1998	Rajyabaquse	Fuzzy logic application of genetic algorithms techniques for adaptive control of nonlinear systems[524]
1998	Wozniak	Knowledge evolution in active database systems via fuzzy constraint management[686]

遗传算法和神经网络都从生物系统的计算中获取灵感。大量的生物学神经结构由遗传过程来决定。因此众多研究人员致力于利用遗传算法来进化神经结构。关于遗传算法和神经网络的论文参阅表 1.14。

表 1.14 遗传算法和神经网络的论文

年 份	作 者	论 文 题 目
1992	Guo	Nuclear power plant fault diagnostics and thermal performance studies using neural networks and genetic algorithms[272]
1992	Rogers	Optimal groundwater remediation using artificial neural networks and the genetic algorithm[539]
1993	Caskey	Genetic algorithms and neural networks applied to manufacturing scheduling[86]
1993	White	GANNet: A genetic algorithm for searching topology and weight spaces in neural network design —the first step in finding a neural network solution[661]
1995	Hanagandi	Process control, optimization, and modeling of chemical systems using genetic algorithms and neural networks[280]
1995	Hashimoto	Study of neural network and genetic algorithms on an application to production planning problems[289]
1995	Igata	Study on estimation of short-time rainy area using optimized neural networks by ggenetic algorithms[314]
1996	Arguelles	Hybrid artificial neural network/genetic algorithm approach to the on-line optimization of electrical power systems[70]
1996	Han	Modeling and optimization of plasma-enhanced chemical vapor deposition using neural networks and genetic algorithms[279]
1996	Jin	Study on optimal neural network design and its application using genetic algorithms[330]
1996	Kermani	On using artificial neural networks and genetic algorithms to optimize performance of an electric nose[355]
1996	Oh	On-line optimization of substrates feeling in hybridoma acell culture using an artificial neural network and genetic algorithm[489]
1997	Chen	Hybrid intelligent system for process modeling control using a neural network and a genetic algorithm[710]

续表

年 份	作 者	论 文 题 目
1997	Kang	Learning evasive maneuvers using evolutionar algorithms and neural networks[344]
1997	Zhou	Study on the convergence of genetic algorithms and its application to structural determination of feed forward neural networks[694]
1998	Fujimoto	Study on parallel processing architecture of neural networks and genetic algorithms[207]
1998	Gong	Evolutionary computation and artificial neural networks for optimization problems and their applications[255]

图像处理和模式识别也是遗传算法界研究的对象。关于遗传算法在图像处理和模式识别领域应用的论文参阅表 1.15。

表 1.15 遗传算法在图像处理和模式识别领域应用的论文

年 份	作 者	论 文 题 目
1993	Tadikonda	Automated image segmentation and interpretation using genetic algorithms and semantic nets[605]
1995	Chintrakulchai	High performance fractal image compression[118]
1996	Bakalis	Encoding the invariant measure of iterated affine transforms with a genetic algorithm[34]
1996	Chunduru	Global and hybrid optimization in geophysical inversion[123]
1996	Matsui	Expression of genetic algorithms to macro adaptation degree and its application to picture processing systems[444]
1996	Swets	Self-organizing hierarchical optimal subspace learning and inference framework for view based recognition and image retrieval[602]
1998	Huang	Detection strategies for face recognition using learning and evolution[310]

调度问题是工业领域中的重要实际问题。由于其具有需求相互矛盾、约束相关和高度的计算复杂性等特点,因此很难用传统优化方法处理。遗传规划和遗传调度在众多研究和应用领域中表现得非常有研究前景。关于遗传算法在规划与调度领域应用的论文参阅表 1.16。

表 1.16 遗传算法在规划与调度领域应用的论文

年 份	作 者	论 文 题 目
1993	Timothy	Optimization of sequencing problems using genetic algorithms[621]
1994	Awadth	Manufacturing process planning model using genetic algorithms[24]
1994	Davern	Architecture for job shop scheduling with genetic algorithms[140]

续表

年份	作者	论文题目
1994	Wright	Genetic algorithm approach to scheduling resources for a space power system[668]
1995	Hoschei	Tabu search/genetic algorithm hybrid heuristic for solving a master production scheduling problem with sequence-dependent changeover times[307]
1995	Kou	Decision support for irrigated project planning using a genetic algorithm[374]
1995	Wang	Scheduling in flowshops with reentrant flows and pallet requirements[653]
1996	Edirisinghe	Optimization of radiotherapy planning using genetic algorithms[170]
1996	Morikawa	Study on scheduling using genetic algorithms[467]
1996	Wall	Genetic algorithm for resource-constrained scheduling[646]
1997	Al-harkan	One merging sequencing and scheduling theory with genetic algorithms to solve stochastic job shop problems[10]
1997	Cheng	Study on genetic algorithms based optimal scheduling techniques[103]
1997	Hocaogle	Multi-path planning using evolutionary computation with specification [302]
1997	Lin	Genetic algorithm-based scheduling system for dynamic job shop scheduling problem[416]
1997	Wang	A genetic algorithm-based approach for subtask matching and scheduling in heterogeneous computing environments and a comparative study of parallel genetic algorithms[652]
1999	Li	Study on hybridized genetic algorithm for production distribution planning problems[411]

关于遗传算法的其他应用论文参阅表1.17。

表1.17 遗传算法的其他应用论文

年份	作者	论文题目
1993	Cowgill	Monte Carlo validation of two genetic clustering algorithms[135]
1993	Elketroussi	Relapse from tobacco smoking cessation: mathematical and computer micro-simulation modeling including parameter optimization with genetic algorithms[176]
1995	Zhuang	Ergonomic evaluation of assistive devices and manual methods for resident-handling tasks in nursing homes[704]
1996	Dighe	Human pattern nesting strategies in a genetic algorithms framework[163]
1997	Hughell	Simulated adaptive management for timber and wildlife under uncertainty[312]
1997	Xie	Genetic Algorithms, the method of regularization and their application to hypocenter location[672]
1998	Freeman	Genetic algorithms: a new technique for solving the neutron spectrum unfolding problem[206]
1998	Graybeal	Pest and resistance management simulation model for the Colorado potato beetle on potatoes[264]

第2章 组合优化问题

2.1 引言

组合优化(combinatorial optimization)研究那些含有有限个可行解的、日常生活中(尤其是工程设计中)大量存在的问题。这其中一个重要并且普遍的应用领域就是考虑如何有效利用稀缺的资源来提高生产力。典型的工程设计问题包括集覆盖、装箱、背包、二次分配、确定最小生成树、机器调度排序与平衡、制造元设计、车辆路径、网络密度、设备定位与布局、旅行推销员分配等。

虽然理论上这种问题的最优解可以通过简单枚举得到,但实际上通常不能实现。特别对于实际规模的问题来说,可行解的数量可能特别巨大。组合优化中最具有挑战性的问题之一就是如何有效处理组合爆炸(combinatorial explosion)。解决这类困难问题的一种重要思路就是采用遗传算法。可以预计未来的几年在这个领域将出现显著的进展。

本章将介绍如何用遗传算法来求解集覆盖、装箱、背包和最小生成树问题。最近提出的应用于其他组合优化问题(如车间作业调度、机器调度排序与平衡、制造元设计、车辆路径、设备布局、网络设计等)的算法将在后续的章节中进行介绍。

2.2 集覆盖问题

集覆盖问题(set-covering problem)是组合优化中的典型问题。问题可以描述为:对于一个 m 行 n 列的 0-1 矩阵,用最小的费用选择一些矩阵的列使其能够覆盖所有的行。设向量 $\boldsymbol{x}$ 的元素 $x_j=1$ 表示列 j 被选中(费用是 $c_j>0$),$x_j=0$ 则表示其未被选中($j=1,2,\cdots,n$)。集覆盖问题可以表示为

$$\min \quad z(\boldsymbol{x}) = \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \geqslant 1, \quad i = 1,2,\cdots,m \tag{2.2}$$

$$x_j \in \{0,1\}, \quad j = 1,2,\cdots,n \tag{2.3}$$

式(2.2)保证每行至少被一列覆盖,式(2.3)是完整性约束。如果所有费用系数 c_j 都相同,则问题称作单一费用集覆盖问题(unicost set-covering problem)。如果式(2.2)为等式约束,上述问题则成为集划分问题(set partitioning problem)。

考虑下面的例子,包括6行6列:

$$c=[10\ \ 15\ \ 11\ \ 10\ \ 8\ \ 2]$$

$$A=\begin{bmatrix}1&1&0&1&0&0\\0&0&1&1&0&0\\1&1&1&0&0&0\\0&0&0&0&1&1\\0&0&1&0&0&1\\0&0&0&1&1&0\end{bmatrix}$$

$$x=[x_1\ \boxed{x_2}\ x_3\ \boxed{x_4}\ x_5\ \boxed{x_6}]$$

$$(a_{ij})=\begin{pmatrix}1&1&0&1&0&0\\0&0&1&1&0&0\\1&1&1&0&0&0\\0&0&0&0&1&1\\0&0&1&0&0&1\\0&0&0&1&1&0\end{pmatrix}$$

$$c=[10\ \boxed{5}\ 11\ \boxed{10}\ 8\ \boxed{2}]$$

图2.1　集覆盖问题的简单例子

这个例子的一个可行解是 $x=[1,0,1,0,1,0]$,费用是29,如图2.1所示。

许多网络问题可以建模为带有特定 $A=(a_{ij})$ 的集覆盖问题,比如Salkin和Saha提出的节点覆盖问题(node-covering problem)[556],Balinski提出的匹配问题(matching problem)[39]和Sallkin和Mathur提出的最大流问题(maximum flow problem)[556]。此外,集覆盖问题在许多现实世界中应用报道见表2.1。

表2.1　集覆盖问题的应用

作　者	应　用
Arabeyre 等(1969)[18]	航线机组成员调度(airline crew scheduling)
Balinski (1965)[39]	开关电路调度(switching circuit scheduling)
Balinski and Quandt (1964)[40]	卡车派遣(trucking dispatching)
Bellmore (1971)[53]	其他例子
Bellmore, Greenberg and Jarvis (1970)[54]	网络攻击与防卫(network attack and defense)
Bellmore (1971)	网络攻击与防卫(network attack and defense)
Busacker and Satty (1965)[82]	图着色(map coloring)
Charnes and Miller (1956)[96]	铁路工作人员调度(railroad crew scheduling)
Cobham, Fridshal and North (1961)[126]	PERT/CPM分析
	开关电路调度(switching circuit scheduling)
	符号逻辑(symbolic logic)
Day (1965)[150]	信息恢复(information retrieval)
Garey Johnson (1979)[208]	资源分配(resource allocation)
Garfinkel (1968)[304]	政治重新划分选区(political redistricting)
Labordere (1969)[349]	电路设计(circuit design)
Levin (1969)[405]	车辆路径(vehicle routing)
Revelle, Marks and Leibman (1970)[535]	设备定位(facilities location)
Root (1964)[540]	符号逻辑(symbolic logic)

续表

作　　者	应　　用
Rubin (1973)[544]	指派问题(assignment problem)
Salverson (1955)[557]	装配线平衡(assembly line balancing)
Steinmann and Schwinn (1969)[587]	装配线平衡(assembly line balancing)
Toregas (1971)[626]	设备定位(facilities location)
Valenta (1969)[636]	资本投资(capital investment)
Walker (1974)[645]	指派问题(assignment problem)

已经证明集覆盖问题是 NP(非确定多项式)完全问题[208]。文献中既有对这类问题的最优解的报道，也有启发式求解的报道。Balas 和 Ho 提出了割平面方法[38]。Fisher 和 Kedia 提出了用于解决最多 200 行 2000 列问题的对偶启发式算法[189]。Beasley 和 Jornsten 结合了 Lagrangian 启发式方法、可行解排除约束、Gomory f-cuts 和改进的分支策略，解决了最多 400 行 4000 列的集覆盖问题[50]。Harche 和 Thompson 开发了一种称作列减少算法(columnsubtraction algorithm)的直接方法，可以解决集覆盖问题的大规模稀疏矩阵的情况[284]。

与其他可变规模组合问题相同，人们最近对于用进化计算方法求解集覆盖问题产生了越来越浓厚的兴趣。Jacobs 和 Brusco 开发了模拟退火算法，并报道在求解最多 1000 行10 000列问题时取得了相当可观的成功[321]。Sen 研究了模拟退火算法和简单遗传算法的性能[570]。Beasley 和 Chu[49]以及 González，Hernández 和 Corne[257]都提出了用遗传算法求解大规模集覆盖问题的方法。

2.2.1 航线机组成员调度问题

由 Salkin 和 Mathur 提出的航线机组成员调度问题(airline crew scheduling problems)[556]是集覆盖问题的一项典型应用。假设需要为 n 个机组成员安排 m 次航班，这些航班必须在一定时段内完成，目标函数是最小化运行费用。由于受到时间、地理位置、机组成员能力(比如不是所有飞行员都有驾驶波音 767 飞机的资格)和其他限制的影响，每位机组成员能够服务的航线的组合可以列写出来。设 k 表示机组成员的编号，t 表示航线的编号(或航线组合的编号)。决策变量 $x_{k(t)}=1$ 表示安排第 k 个机组成员到第 t 次航线组合上。系数 $a_{ik(t)}=1$ 意味着第 t 个航线组合的第 i 次航程(flight leg)(比如 Flight 347，从 Cleveland 到 Chicago)可以由第 k 个机组成员完成。有如下的约束：

$$\sum_k \sum_t a_{ik(t)} x_{k(t)} \geqslant 1, \quad i = 1,2,\cdots,m \tag{2.4}$$①

① 译者注：式(2.4)表明，每 1 次航程都至少有 1 个机组成员来完成。

设 $c_{k(t)}$ 表示第 k 个机组成员的第 t 次航线组合的费用。问题就是安排最优的机组成员使得总费用最低。

表2.2表示了4个机组成员和5次航程的简单例子。机组成员1可以驾驶航程1,2,4和5,或1,3和4,或1和3。机组成员2可以驾驶航程1,2和3,或1和2,等等。在给出第 k 个机组成员的第 t 次航线组合的费用之后,就需要确定最优的航线机组成员调度方案。

表2.2　航线机组成员调度的例子

机组成员 k	1			2		3		4		
航线组合 t	1	2	3	1	2	1	1	1	1	
变量 $x_{k(t)}$	$x_{1(1)}$	$x_{1(2)}$	$x_{1(3)}$	$x_{2(1)}$	$x_{2(2)}$	$x_{3(1)}$	$x_{3(2)}$	$x_{4(1)}$	$x_{4(2)}$	
航程 i	$a_{i1(1)}$	$a_{i1(2)}$	$a_{i1(3)}$	$a_{i2(1)}$	$a_{i2(2)}$	$a_{i3(1)}$	$a_{i3(2)}$	$a_{i4(1)}$	$a_{i4(2)}$	
1	1	1	1	1	1	0	0	0	0	$\geqslant 1$
2	1	0	0	1	1	1	1	1	1	$\geqslant 1$
3	0	1	1	1	0	0	1	1	0	$\geqslant 1$
4	1	1	0	0	0	1	0	0	0	$\geqslant 1$
5	1	0	0	0	0	0	1	0	0	$\geqslant 1$

2.2.2　遗传表示

有两种表示(representation)集覆盖问题的方法:(1)基于列的表示(column-based representation),(2)基于行的表示(row-based representation)。

基于列的表示　基于列的表示表示了集覆盖问题本质上的0-1变量,是显而易见的表示方式。比如对于 n 列的问题,可以用 n 位的二进制字符串来作为染色体的结构。在第 i 位的1表示解中包含列 i。图2.2具体说明了表示方法。

列(基因)	x_1	x_2	x_3	x_4	x_5	x_6
位串	1	0	1	0	1	0

图2.2　基于列的表示

个体的适应值 $f(\boldsymbol{x})$ 可以简单由下式计算:

$$f(\boldsymbol{x}) = \sum_{j=1}^{6} c_j x_j \tag{2.5}$$

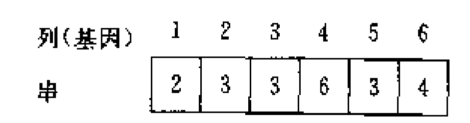
初始种群可以随机选取。注意用基于列表示的个体不能保证都是可行的,这意味着并非所有行都被覆盖了。因此需要某些矫正机制来将不可行表示转换为可行表示。通常有3种处理不可行解的方法。

一种方法是拒绝繁殖过程中产生的所有不可行解,这可能导致无法获得可行解。另一种方法是应用罚函数来惩罚不可行解的适应值。Bäck 对在集覆盖问题中采用罚函数

和修补启发式方法进行了比较研究[28]。第三种方法是修补在繁殖过程中产生的不可行染色体。Beasley 和 Chu 倾向于一种使用附加启发式算子的修补方法，这种方法可以将不可行解转换为可行解[49]。

基于行的表示 确保可行性的问题(所有行都被覆盖)可以由基于行的表示来解决。一种可行的表示方法是让个体染色体的长度与给定问题行的数量相同。这种表示中，每一基因的位置对应着某行，而每一基因的值则对应着一个覆盖该行的列。

图 2.3 表示了这样一个例子，其中行 1 被列 2 覆盖，行 2 被列 3 覆盖等。采用这种表示，在杂交和变异过程中始终可以保持可行性。但是评价染色体成为不明确的工作，原因在于同一解可以表示成不同的形式，而相同的形式可能得出不同的适应值(这由染色体如何表示来决定)①。由于同一列可能在超过一个基因位置中出现，就需要改进解码方法来从染色体的表示中抽取惟一的列集合(重复的列仅考虑一次)并进行适应值评价。González, Hernández 和 Corne 采用了这种表示方案[257]。



列(基因)	1	2	3	4	5	6
串	2	3	3	6	3	4

图 2.3 个体基于行的表示

2.2.3 遗传算子

Beasley和Chu提出了一种通用的基于适应值的杂交算子，称作熔合算子(fusion operator)[49]。这种算子既考虑父代解的结构，又考虑他们的适应值。与传统杂交算子通常生成两个后代不同，而熔合算子仅生成一个后代。

设 P_1 和 P_2 表示父代染色体串，f_{P1} 和 f_{P2} 分别表示两个父代的适应值，C 表示子代染色体串。融合算子如下所示：

熔合算子过程

第 1 步：$i=1$

第 2 步：如果 $P_1[i]=P_2[i]$，则 $C[i]:=P_1[i]:=P_2[i]$

第 3 步：如果 $P_1[i]\neq P_2[i]$，则

(3.1) 以概率 $p=f_{P2}/(f_{P1}+f_{P2})$ 让 $C[i]:=P_1[i]$②

(3.2) 以概率 $1-p$ 让 $C[i]:=P_2[i]$

第 4 步：如果 $i=n$，停止；否则 $i=i+1$，返回第 1 步。

---

① 译者注：用 2.1 的 **A** 矩阵和图 2.3 的表示可以说明这句话。基于行表示的染色体[2 3 3 6 3 4]可以翻译为基于列表示的染色体[0 1 1 1 0 1]，但也可以翻译为基于列表示的染色体[0 1 1 1 1 1]；反过来，基于列表示的染色体[0 1 1 1 0 1]可以翻译为基于行表示的染色体[2 3 3 6 3 4]，但也可以翻译为基于行表示的染色体[4 3 3 6 3 4]。也就是说，基于行表示的染色体和基于列表示的染色体不是一一对应的。而我们知道，可行的基于列表示的染色体与集覆盖问题的解之间是一一对应的，因此基于行表示的染色体存在适应值评价的问题。

② 译者注：由于是求最小化问题，因此适应值高个体的基因被选为子个体基因的概率低。

变异通常是以小概率翻转个体中的某一位。Bäck 提出 $1/n$ 作为最优变异率的下界(n 是染色体的长度[26])。Beasley 和 Chu 提出可变的变异方案[49]。变异的位的数量由下式决定：

$$\frac{m_f}{1+\exp[-4m_g(c-m_c)/m_f]} \tag{2.6}$$

其中 c 是产生的子代个体的数量①，m_f 表示最终稳态的变异率，m_c 表示当变异率是 $m_f/2$ 时产生的子代个体数量，m_g 表示 $c=m_c$ 时的斜率。m_f 的值用用户指定，m_c 和 m_g 的值由特定问题的遗传算法收敛所需要的速率来确定②。

Beasley 和 Chu 采用了基于列的表示[49]。为了避免个体的不可行性，他们给出了一种启发式算子。该算子包含两个主要成分：修补给定个体的不可行性和执行附加的局部搜索以试图提高遗传算法的效率。

确保每个个体可行的过程包括：(1)确定所有没被覆盖的行，(2)添加必要的列使得所有行都被覆盖。一旦添加了列使得个体成为可行解，就应用局部搜索过程来消除个体中冗余的列。冗余列就是从个体中去掉后依然保持个体可行性的列。

不失一般性，假设列按照费用升序排列。具有相同费用的列按照他们覆盖行的数量进行降序排列。为了叙述过程，采用下述符号：

I：所有行的集合

J：所有列的集合

α_i：覆盖行 $i(i\in I)$的列的集合

β_j：被列 $j(j\in J)$覆盖的行的集合

S：解中列的集合

U：未被覆盖的行的集合

w_i：S 中覆盖行 $i(i\in I)$的列的数量

修补不可行解的过程描述如下：

启发式算子过程

第1步：初始化 $w_i:=|S\cap\alpha_i|,\forall i\in I$③

第2步：初始化 $U:=\{i|w_i=0,\forall i\in I\}$

第3步：对于 U 中每一行 i(按照 i 的升序)

(3.1) 在 α_i 中寻找第一个最小化 $c_j/|U\cap\beta_j|$ 的列(按照 j 的升序)④

---

① 译者注：参见后面叙述的算法过程。

② 译者注：早期变异率较小(c 小导致式(2.6)分母值较大)。随着子个体数量的增加到 m_c，变异率为 $m_f/2$。随后，变异率逐渐增加到 m_f。

③ 译者注：w_i 表示当前解中覆盖第 i 行的列的数量。

④ 译者注：$|U\cap\beta_j|$ 表示被第 j 列覆盖但目前尚未被当前解覆盖的行的数量。最小化 $c_j/|U\cap\beta_j|$ 表示希望第 j 列费用低，而且在当前解中加入第 j 列后覆盖尚未被覆盖的行数多。

(3.2) 将第 j 列添加到 S，令 $w_i:=w_i+1, \forall i\in\beta_j$，令 $U:=U-\beta_j$

第 4 步：对 S 中的每一列 j(按照 j 的降序)，如果 $w_i\geqslant 2, \forall i\in\beta_j$，则令 $S:=S-j$ 以及 $w_i:=w_i-1, \forall i\in\beta_j$①

第 5 步：S 现在就是没有冗余列的可行解。

第 1 和第 2 步确定未被覆盖的行。第 3 和第 4 步是“贪心”的启发式方法。第 3 步中首先考虑最小费用的列，而第 4 步中首先考虑去除最高费用的冗余列。

2.2.4 遗传算法

Beasley 和 Chu 提出的用遗传算法解集覆盖问题的方法归纳如下：

遗传算法解集覆盖问题的过程

第 1 步：随机产生 N 个解构成初始种群。令 $t:=0$。

第 2 步：采用二进制竞争选择从种群中选出两个解 P_1 和 P_2。

第 3 步：采用熔合杂交算子组合 P_1 和 P_2 生成新解 C。

第 4 步：变异 C 中 k 个随机选出的列，其中 k 由可变的变异方案确定。

第 5 步：采用启发式算子确保 C 可行并去除冗余列。

第 6 步：如果 C 与种群中任意解相同，返回第 2 步。否则令 $t:=t+1$，到第 7 步。

第 7 步：从种群中随机选出一个高于平均适应值的个体并用 C 替代(稳态复制方法)。②

第 8 步：重复第 2 到第 7 步直到产生 $t=M$ 个不重复个体。从种群中选出具有最小适应值的个体作为最优解。

2.2.5 计算经验

Beasley 和 Chu 提出的测试问题的规模从 200×1000 到 1000×10 000[49]。这些问题可以从 OR-Library(电子邮件地址：or.library@ic.ac.uk)获得[48]。这些问题中，有的知道最优解的值，而有的仅知道历史上的最好解。根据 Beasley 和 Chu 的计算结果，遗传算法可以得到比那些历史上最好解好得多的近似最优解。

在他们的试验中，每个测试问题采用了 10 次计算，每次计算使用不同的随机种子。每次计算当产生 $M=100\ 000$ 个不重复解时终止。所有问题的种群规模 N 设为 100。表 2.3给出了一些计算结果(具有相同规模的每一问题的最好情况和最坏情况)。

最优解的值由文献[50]给出，历史上最好解的值由文献[32]给出。平均百分偏差(σ)由 $\sum_{i=1}^{10}(S_{Ti}-S_o)/10S_o\times 100\%$ 计算。其中 S_{Ti} 是第 i 次试验的最小(最好)解值，而 S_o 则是

① 译者注：意思是如果当前解中存在第 j 列，它所覆盖的所有行都至少被当前解中的 2 列以上覆盖。

② 译者注：由于是求最小化问题，因此适应值高的个体被选作替换对象。

最优解值。负值意味着遗传算法得到了比历史上最好解更优秀的结果。求解时间(用CPU秒度量)是遗传算法用于搜索最终最优解的时间,而执行时间(用CPU秒度量)则是用遗传算法产生100 000个不重复子个体的时间。

表2.3 采用遗传算法优化集覆盖问题的计算结果

问题规模 $m\times n$	密度(%)	最优解或历史最好解	10次计算最好解	平均 σ(%)
200×1000	2	512	512	0.00
		641	641	0.33
200×2000	2	279	279	0.00
		226	228	0.88
200×1000	5	131	131	0.00
		146	146	0.14
300×3000	2	236	236	0.00
		253	253	0.79
300×3000	5	72	72	0.00
		80	80	0.00
400×4000	2	215	215	0.05
		243	243	1.40
400×4000	5	60	60	0.00
		72	72	0.28
500×5000	10	28	28	0.00
		27	27	2.60
500×5000	20	14	13	−2.14
		14	14	0.00
1000×10 000	2	158	155	−1.08
		168	168	0.83
1000×10 000	5	60	59	−1.50
		55	55	0.18

2.3 装箱问题

装箱问题(bin-packing problem)要将 n 个物品装入许多箱子(最多 n 个箱子)。每个物品有重量($w_i>0$),每个箱子有重量限制($c_i>0$)。问题是寻找最优的将物品分配到箱子的方案,从而使每个箱子中物品的重量之和不超过其限制,而使用的箱子数量最少。该问题可以表示为

物品	1	2	…	n
重量	w_1	w_2	…	w_n
重量限制	c_1	c_2	…	c_n

其中对于 n 个箱子来说，重量 w_i 和重量限制 c_i 是正实数($i=1,2,\cdots,n$)。

通常，所有箱子有相同的重量限制($c>0$)。显然，将每个物品放入一个箱子是一个可行解，但不是最优解。装箱问题的数学表示如下[441]：

$$\min \quad z(y)=\sum_{i=1}^{n} y_i \tag{2.7}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_{ij} \leqslant c y_i, \quad i \in N=\{1,2,\cdots,n\} \tag{2.8}$$

$$\sum_{i=1}^{n} x_{ij}=1, \quad j \in N \tag{2.9}$$

$$y_i=0 \text{ 或 } 1, \quad i \in N \tag{2.10}$$

$$x_{ij}=0 \text{ 或 } 1, \quad i,j \in N \tag{2.11}$$

其中，$y_i=1$ 表示箱子 i 被装入物品，反之则表示箱子 i 空着。$x_{ij}=1$ 表示物品 j 放入箱子 i，反之表示物品 j 未放入箱子 i。

装箱问题是许多具有重要意义的实际优化问题的基础[182]。在建筑中经常需要从长度一定的棍子上切割不同长度的棒和管。电气布线中的电线来自长度一定的线卷。贴墙纸需要从给定长度的纸卷中得到。具有相同宽度、不同长度的石料需要用平板架运送到建筑工地。在金属制造工业中，钢片需要从大块钢片中切除。物品的尺寸不一定是几何尺寸。比如，电子工业中不同长度(字节)的微代码程序需要存储至微处理器中具有固定尺寸的存储器中。在运输工业中，卡车具有给定的最大承重，通常将不同负荷的重量作为考虑对象。在娱乐工业中，歌曲集被发布在给定容量的媒质(光盘、磁带)上。在作业管理中，不同执行时间的任务需要在满足完成所有任务时间限制的前提之下被分配到不同的工人。重要的生产流水线平衡(沿着生产流水线将任务分配到工作站)问题就是带有附加优先约束的装箱问题。装箱问题是容量限制的工厂选址问题的特例之一，该问题将于7.5节讨论。

2.3.1 启发式算法

装箱问题属于NP-难问题[208]。目前还没有能够在多项式时间内求得最优解的算法。针对该问题仅提出了一些*启发式算法*(heuristic algorithms)。但大多数启发式算法以贪心方法为特点，采用了某些简单规则，比如次优配合、优先配合或最佳配合。Garey 和 Johnson 指出，简单启发式算法的结果可以不差于(但也不好于)最优解乘以一个相当小的系数[208]。

次优配合启发式方法(next-fit heuristic) 装箱问题最简单的启发式方法是次优(NF)算法。第1个物品放入第1个箱子，然后是根据下标上升的顺序放入第2，…，n 个物品。如果当前箱子的容量允许，则每个物品放入当前箱子，否则放入一个新的箱子。于

是新箱子成为当前箱子。该算法的复杂性是 $O(n)$。对于给定的问题 I，容易证明算法给出的解值 NF(I)满足下面约束：

$$NF(I) \leqslant 2z(I) \tag{2.12}$$

其中 $z(I)$代表最优解值。

优先配合启发式方法(first-fit heuristic) 优先配合(FF)算法也根据下标上升的顺序考虑物品。但将每个物品放入其能够放入的最小下标的已初始化的箱子。如果当前物品无法放入任何已初始化的箱子，则放入一个新箱子。Johnson 等人证明，对于所有装箱问题的情况都有[331]：

$$FF(I) \leqslant \frac{17}{10}z(I) + 2 \tag{2.13}$$

最佳配合启发式方法(best-fit heuristic) 最佳配合(BF)算法从 FF 算法改进而来，它将当前物品放入具有最小剩余容量的可行箱子中。该算法摒弃了有利于最小下标箱子的思想。Johnson 等人证明 BF 满足与 FF 相同的最坏情况边界[331]。FF 和 BF 的时间复杂性都是 $O(n \log n)$。

其他启发式方法 如果物品按照重量降序排列($w_1 \geqslant w_2 \geqslant \cdots \geqslant w_n$)，然后采用 NF，FF 或 BF 得到的算法相应称作次优配合降序(next-fit decreasing，NFD)，优先配合降序(first-fit decreasing，FFD)和最佳配合降序(best-fit decreasing，BFD)。这些算法的时间复杂度都是 $O(n \log n)$。这些算法通常用于测试新提出的用于装箱问题的算法的有效性。通常他们都嵌入遗传算法来增强遗传搜索能力，从而为装箱问题寻找到更接近最优解的答案。

2.3.2 遗传表示

装箱问题已经提出了 3 种表示方法：(1)基于箱子的表示，(2)基于物品的表示，(3)基于群体的表示。

基于箱子的表示(bin-based representation) 最直接的方法就是将物品的从属关系进行编码。在基于箱子的表示中，基因的位置表示物品，基因的值表示该物品放入的箱子。比如说，染色体 1 4 2 3 5 2 就形成了一个解，其中第 1 个物品放入箱子 1，第 2 个放入箱子 4，第 3 个放入箱子 2，第 4 个放箱子 3，第 5 个放入箱了 5，第 6 个放入箱子 2。基于箱子的表示可以用图 2.4 进行说明。

	1	2	3	4	5	6	← 物品
染色体	1	4	2	3	5	2	← 箱子

图 2.4 物品从属关系的表示

这种表示的优点是染色体具有恒定的长度(与物品的数量相同)，这样就可以采用标准遗传算子。然而，正如 Falkenauer[181]指出的那样，这种编码具有许多缺点。首先，这种编码

高度冗余。事实上，费用函数仅依赖与箱子的数量，而与箱子中物品的数量无关。比如，1 2 3 1 4 4和3 1 4 3 2 2就是对问题相同解的不同编码。也就是说，第1和第4个物品放入同一个箱子，第5和第6个物品放入同一个箱子，第2和第3个物品分别被放入两个箱子。

这种方式编码的冗余度(也就是同一问题中相同解编码出不同染色体的数量)随着箱子数量的增加而指数上升，也就是间接地随问题的规模指数上升。因此遗传算法搜索的空间就比问题本身的解空间大得多。因此遗传算法的搜索能力受到严重削弱。

其次，这种编码会使某基因的含义依赖于其他基因，这种情况在采用标准杂交算子时是非常不希望发生的。比如，图2.5表示了将标准2点杂交算子应用于两个染色体并得到其后代的过程。通常两个相同的父代杂交产生的后代将与其父代完全相同。图2.5的两个父代完全相同，原因在于他们都是对同一解的编码。因此杂交算子应该产生与他们相同的个体。然而杂交产生的后代却与其父代所代表的解完全不同(后代使用了两个箱子，而不是四个)。换句话说，当关于染色体的模式得以在标准编码/杂交中进行传递时，染色体所代表的关于问题解的含义却在杂交过程中丧失了。

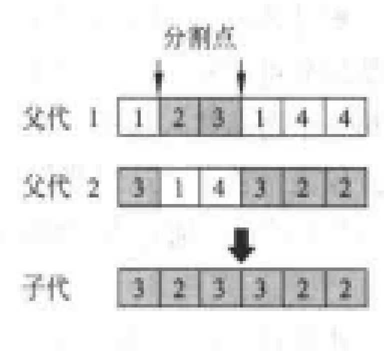


图2.5 杂交操作

这种编码的第三个缺点是可能由于一个箱子被放入了超过其重量限制的过多物品而产生不可行解。

基于物品的表示(object-based representation) 另一种关于装箱问题的表示方式是对物品的排列进行编码，然后应用解码器得到其对应的解。假设有6个物品需要装箱，编码就是数字1到6的排列。比如下面就代表了一条染色体：

1 2 3 4 5 6

这种编码同样有多方面的缺点。首先，这种编码也是高度冗余的。事实上，假设上述染色体可以进行如下的划分：

1 2 3 | 4 5 | 6

就是说，有3个箱子，一个装物品1到3，第2个装物品4和5，第3个装物品6。于是同一箱子里的物品的任意排列均产生新的染色体，而其代表的解却相同。比如：

3 2 1 | 4 5 | 6

或

4 5 | 2 1 3 | 6

同时，这种编码的冗余度随着问题规模的增加而指数上升，遗传算法的能力受到严重影响。

其次，由于事实上解码器是根据给定的染色体排列从左到右将物品放入箱子从而得到问题的解，因此某物品的放入的箱子号严重依赖于染色体的“头”(染色体最左边位置的物品)。然而，给定基因的位置与问题的费用函数没有任何关系。结果就是，这种表示方

法无法在进化过程中维持从父代继承的信息。与基于箱子的表示不同，这种方法从不产生不可行解。

基于群体的表示(group-based representation)　上述两种表示方法对于装箱问题这样的群体问题不适合，原因在于这些染色体是面向项(item)的，而不是面向群体(group)的。简言之，由于装箱问题的费用函数依赖于箱子中物体的群体，上述编码不适合用作这类问题的费用函数优化。对于装箱问题，更好的表示方法需要包含两个部分。第1部分提供了关于哪个物品属于哪个箱子(群体)的信息。第2部分对使用的箱子进行编码。基于这样的考虑，Falkenauer[181]提出了一种表示方法，带有物品的群体表示。这种方法用一个基因表示一个箱子。

更具体地说，让我们考虑2.3.2节提出的染色体。从1到6对物品进行编码，染色体物品部分可以写作1 4 2 3 5 2。这标志着第1个物品放入箱子1，第2个放入箱子4，第3和第6个物品放入箱子2，第4个物品放入箱子3，第5个物品放入箱子5。染色体的群体部分仅表示箱子。这里我们采用字母而不是整数来表示箱子(比如，上述染色体可以表示为ADBCEB)。通过查询物品部分，我们可以搞清楚群体的名字代表的含义，即

$$B=\{3,6\},\ E=\{5\},\ C=\{4\},\ D=\{2\},\ A=\{1\}$$

此外，包含两个部分的染色体的集成可以由图2.6来表示。这种编码使得基因既表示物品又表示群体(箱子)①。这种表示的原理是：对于装箱问题来说，群体是有意义的积木块(也就是最小的解片断，这些片断可以按照对其所属的解所期望的质量来转运信息)。这种表示的关键之处就是遗传算子对于染色体的群体部分进行操作，其物品部分仅用于判定由哪些物品形成该群体。特别指出一点，这种表示意味着算子需要处理不同长度的染色体。

3,6	5	4	2	1	←物品
B	E	C	D	A	←箱子

图2.6　物品的群体表示

2.3.3　遗传算子

本节中我们将重点介绍由Falkenauer[182]提出的基因群体表示的遗传算子。

杂交　正如上一节指出的那样，基于群体的表示包含两个部分：箱子和物品的群体。因此杂交需要处理可变长度的染色体，这种染色体用基因表示箱子。这种杂交过程如下所述，可以用图2.7说明。

杂交过程

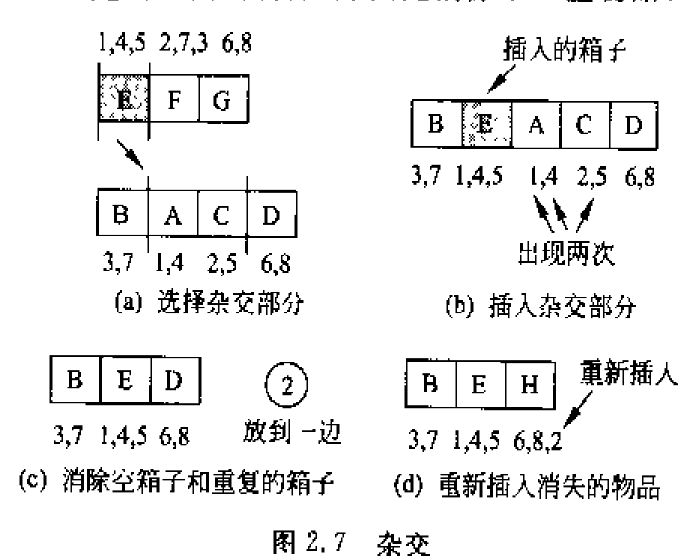
第1步：随机选择两个杂交位置，对每个父代选定杂交部分。

第2步：将第1个父代杂交部分的内容插入到第2个父代第1个杂交位置之前。由于杂交对染色体的部分群体进行操作，这就意味着从第1个父代插入一些群体(箱

① 译者注：群体的意思是将某个箱子中的物品的集合用一个字母来表示。

子)到第 2 个父代中。

第 3 步：从产生的后代中原有的箱子中去掉所有重复出现的物品，使得这些物品原先的从属关系让位于“新”插入的箱子。因此产生的后代中的某些群体发生了改变。他们不再包含与先前相同的物品，原因是消除了一些物品。



(a) 选择杂交部分 (b) 插入杂交部分

(c) 消除空箱子和重复的箱子 (d) 重新插入消失的物品

图 2.7 杂交

第 4 步：如果必要，根据问题的约束和待优化的费用函数来调整新产生的箱子。在本阶段，可以采用如 FFD 等局部搜索算法。

第 5 步：改变两个父代的角色并重新应用第 2 步到第 4 步生成第 2 个子代。

变异 装箱问题的变异算子必须针对群体(箱子)而不是物品进行操作。至于变异过程，算子的实现细节依赖于现有的特定群体问题。但可以指出两条一般性的策略：或启用一个新的箱子，或消除一个已经使用的箱子。如果变异后解中缺少某些物品，可以采用 FF 或 FFD 启发式方法来按照随机的顺序将其重新放入箱子。

2.3.4 适应值函数

用遗传算法来解装箱问题存在两种评价目标函数的方法：(1)最小化使用的箱子数量，(2)最小化使用的箱子数量同时尽量装满所有使用的箱子。Falkenauer 和 Delchambre[184]对装箱问题提出了下面的评价函数(属于上述第二种方法)：

$$f_{BPP}=\frac{\sum_{i=1}^{N}(F_i/C)^k}{N} \tag{2.14}$$①

其中 N 是解中使用的箱子数量，F_i 是第 i 个箱子中所有物品的重量之和(箱子的填充程度)，C 是箱子的重量限制，k 是常数($k>1$)。常数 k 表示了对装得满的箱子的重视程度。

① 译者注：遗传算法优化的目标是最大化 f_{BPP}。一方面，使用的箱子数量越少，f_{BPP} 越大。另一方面，F_i 越接近 C，f_{BPP} 越大。

k 越大，装得满的箱子比一般填充的箱子受到的重视就越大。根据他们有限的计算经验，$k=2$ 得到的结果较好。

2.3.5 初始化种群

装箱问题有两种产生初始种群的方法：随机的方法和启发式的方法。第一种方法随机产生所有的初始个体。但用FF启发式方法产生装箱问题所有的个体也比较容易。这就意味着可以用FF启发式规则将所有物品放入箱子中。这种方法可能由产生箱子(基因)数相差很大的个体组成的种群。

2.3.6 计算经验

Bilchev[710]和Falkenauer[181]报道了一些计算结果。Bilchev采用基于物品的表示来处理装箱问题。表2.4表示了遗传算法方法和FFD启发式方法的有效性比较。

表2.4　装箱问题的计算结果

问题规模	最优解	FFD	GA
30	9	11	9
60	18	22	18
90	27	33	27
120	36	44	36
150	45	55	45

Falkenauer[181]采用基于群体的方法来处理装箱问题。测试数据按照下面规则创建：首先产生代表完美装箱($f_{BPP}=1$)的物品，然后从这些物品中减去一个箱子总重量限制的 $p\%$。举例来说，如果 $p=10\%$，箱子重量限制 $C=255$，物品的总重量比完美装箱时箱子的重量限制小25.5。这样测试就反映了算法接近最优解的能力，而不是寻找完美装箱的能力。事实上，后者测试的是最优性，这与在合理的时间内解决NP完全决策问题(这种问题是现今算法不能解决的)是等价的。

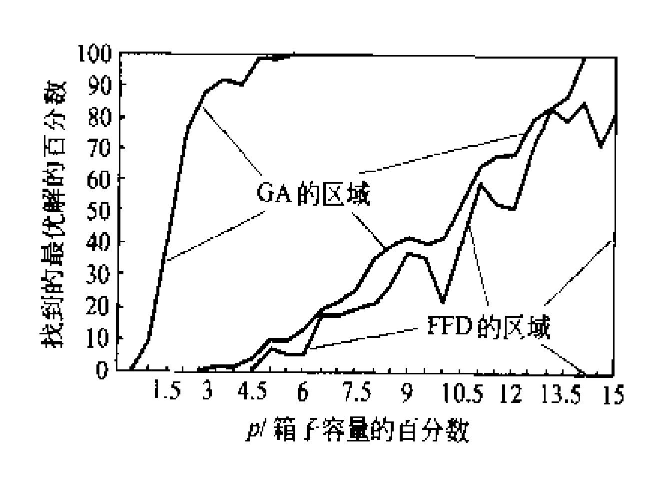


图2.8　FFD启发式方法和遗传算法的相对性能比较

64个物品的计算结果如图2.8所示。该图表明当 $k=2$ 时 p 与分别由FFD启发式算法和遗传算法成功找到问题最优解的比例的函数关系。为了计算出找到最优解的比例，根据前述过程针对每个 p 值(从 C 的1.5%到15%)创建了50种测试情况。

2.4 背包问题

在过去的10年中，背包问题(knapsack problem)吸引了理论研究人员和实际工作者的注意力，因而得到了深入的研究。理论方面研究兴趣来自于该问题简单的结构，这种特点既可以深入探索许多组合特性，又可以通过解决一系列背包子问题来最终求解更为复杂的优化问题。从实践的角度来看，这些问题可以表述许多工业场合的应用，最典型的应用包括资本预算、货物装载和存储分配等[441,478]。

假设我们需要从许多物品(通常称作项目)中选择一些来填充一个背包。存在 n 个不同的项目可以使用，每个项目 j 具有重量(weight) w_j 和费用(cost) c_j。背包可以承重的上限是 W。问题是如何寻找项目的最优子集从而在满足背包承重约束的基础上最大化总费用。费用、重量和承重是正整数。

考虑8个项目的背包问题作为例子。每个项目的重量和费用在表2.5中定义。背包的承重定义为 $W=140$。问题额可以用图2.9清晰地说明。

表 2.5 背包问题的重量和费用

项目号	1	2	3	4	5	6	7	8
重量	35	50	30	15	10	35	25	40
费用	40	60	25	20	5	60	40	25

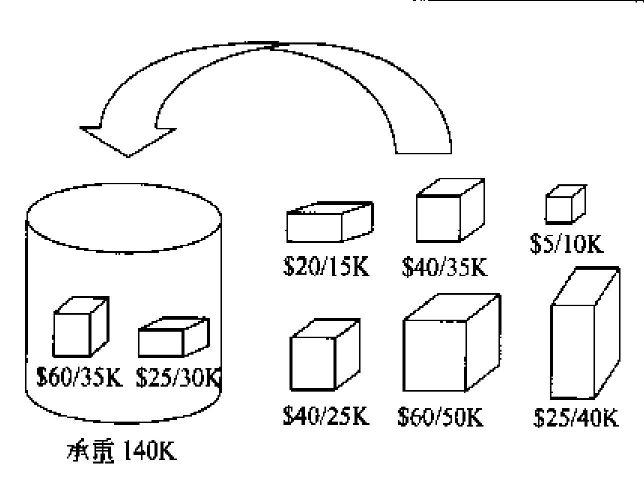


图 2.9 背包问题

设 x_j 为二进制变量。如果项目 j 被放人背包，则 $x_j=1$，否则 $x_j=0$。背包问题的数学描述如下：

$$\max \quad \sum_{j=1}^{n} c_j x_j \tag{2.15}$$

$$\text{s. t.} \quad \sum_{j=1}^{n} w_j x_j \leqslant W \tag{2.16}$$

$$x_j \in \{0,1\}, \quad j = 1,2,\cdots,n \tag{2.17}$$

这种问题就是所谓0-1背包问题(0-1 knapsack problem),带有一个简单约束的纯整数规划,是一类非常重要的整数规划问题。有许多背包问题的变形,比如多选择背包问题(multiple choice knapsack problem)、有界背包问题(bounded knapsack problem)、无界背包问题(unbounded knapsack problem)和多约束背包问题(multiconstrained knapsack problem)[441]。

背包问题已经被证明是NP难的[208]。Salkin和De Kluyver通过将整数线性规划问题转换为背包问题(这种方法当时显得非常有前途)提出了许多工业应用和优化结果[554]。Martello和Toth考虑了0-1背包问题的直接算法及其平均计算性能[439]。该项研究由martello和Toth[440]延伸到其他线性背包问题并提出了近似算法。Dudzinski和Walukiewicz分析了求解Lagrangian和线性规划松弛的对偶方法[168]。

与其他组合优化问题类似,许多研究人员常用遗传算法(GAs)来求解背包问题。其中包括Olsen提出的遗传算法[491],Kubota和Fukuda采用的二进制表示[377],Hinterding采用的次序表示[298]以及Gordon和Whitley采用的变长表示[259]。本节将介绍Gen等人提出的用遗传算法求解多选择背包问题[220]和Raidl提出的用遗传算法求解多约束背包问题[522]。

2.4.1 多选择背包问题

问题描述 多选择背包问题定义为有附加约束的背包问题,该问题带有互不相关的多选择约束。该问题的一般性描述如下:有一个承重有限的背包。将要放入背包的物品被分为相互排斥的若干类。每类中有若干不同的项目。问题就是从每类中选择一个项目使得项目总重量在满足背包承重约束的前提之下最小化费用。问题如图2.10所示。

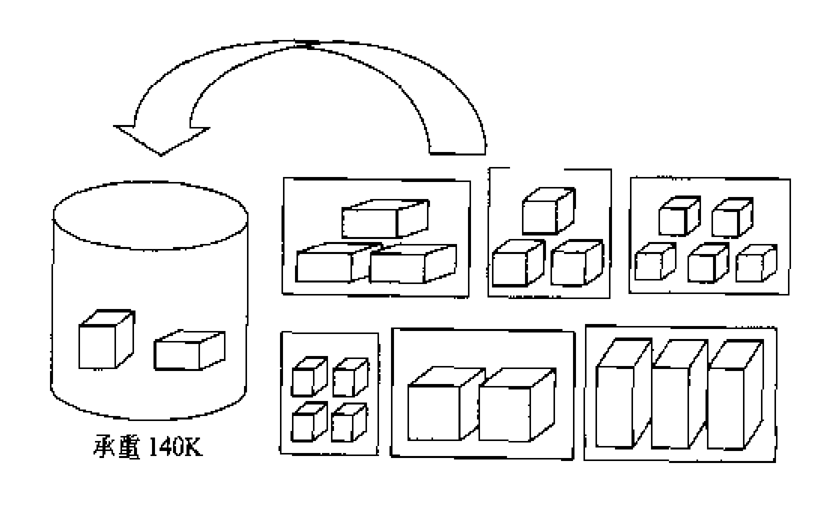


图2.10 多选择背包问题

多选择背包问题的数学描述如下：

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} c_{ij}x_{ij} \tag{2.18}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} w_{ij}x_{ij} \leqslant W \tag{2.19}$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, \quad i = 1,2,\cdots,m \tag{2.20}$$ ①

$$x_{ij} \in \{0,1\}, \quad \forall i,j$$ ②

其中 i 是类的下标，j 是每类中项目的下标，n_i 是第 i 类中项目的数量，c_{ij} 是第 i 类中第 j 个项目的费用，m 是类的数量，w_{ij} 第 i 类中第 j 个项目的重量，W 是背包的承重。采用双下标使得变量的 m 个多选择集相互排斥。约束(2.20)称作*多选择约束*(multiple-choice constraints)或*广义上界*(generalized upper bounds, GUBs)。

该问题是某种广义的指派问题，属于 NP 难问题。解决该问题最成功的方法是分支定界算法，这种算法采用了线性规划[291]、Lagrangian 松弛[243,601]或 Lagrangian 松弛变形[46]来定界。

遗传表示(representation) 许多研究人员应用遗传算法解决了背包问题，他们的方法可以粗略分成 3 种[219]：(1)*二进制表示*(binary representation)方法，(2)*次序表示*(order representation)方法，(3)*可变长度表示*(variable-length representation)方法。

但对于多选择背包问题来说，某种排列对于编码来说更加自然也更为有效[220]。一个染色体由 m 个基因组成，对应着 m 个类。第 i 个基因从相互排斥集合 N_i 中选取整数，N_i 是第 i 类所包含的项目的集合($\{1,2,\cdots,n_i\}$)。于是基因的位置用来表示物品的类，而基因的值则用来表示从类中选出的项目。定义指示变量 y_i 如下：

$$y_i = j \qquad \text{如果 } x_{ij} = 1,\ j \in N_i,\ i = 1,2,\cdots,m \tag{2.21}$$

遗传表示可以形式上定义为如图 2.11 所示的样子。

考虑下面的例子

$$\begin{aligned}
\min \quad & x_{11} + 4x_{12} + 5x_{13} + 7x_{14} + 7x_{21} + 9x_{22} + 10x_{23} + 5x_{31} + 6x_{32} + 11x_{33} \\
\text{s.t.} \quad & 10x_{11} + 8x_{12} + 5x_{13} + 4x_{14} + 6x_{21} + 5x_{22} + 3x_{23} + 8x_{31} + 6x_{32} + 4x_{33} \leqslant 16 \\
& x_{11} + x_{12} + x_{13} + x_{14} = 1 \\
& x_{21} + x_{22} + x_{23} = 1 \\
& x_{31} + x_{32} + x_{33} = 1
\end{aligned}$$

图 2.12 表示了这种编码方式的一个例子，对应着解 $x_{12}=1$，$x_{23}=1$，$x_{33}=1$。容易验证

① 译者注：该式的含义是每类中只能选出一个项目放入箱子中。

② 译者注：x_{ij} 的含义是：如果第 i 类的第 j 个项目被放入箱子，则 $x_{ij}=1$，否则 $x_{ij}=0$。

这是可行解。



图 2.11 排列编码　　　　　图 2.12 排列编码的例子

杂交和变异 简单地说，可以为遗传操作采用均匀杂交和随机扰动。在随机扰动中，被选中的基因 y_i 用集合 N_i 中随机选出的一个整数替代。

评价和选择 将惩罚项添加到适应值函数中，从而迫使遗传搜索从解空间的可行和不可行两个方向接近最优解。评价函数包含两项：总费用和对不可行的惩罚。总费用就是计算出来的目标函数。设 v_k 是当前种群的第 k 个染色体，x_{ij}^k 是对应的决策变量，则第 k 个染色体的费用计算如下：

$$f_k(v_k) = \sum_{i=1}^{m}\sum_{j\in N_i} c_{ij}x_{ij}^k$$

一般来说，由于所选项目的总重量超过了背包的承重，编码可能对应着不可行解。惩罚系数 p_k 与解超出的重量成比例，如下式所示：

$$p_k = \begin{cases} 0, & \text{如果约束(2.19)得到满足} \\ \alpha_0\left(\sum_{i=1}^{m}\sum_{j\in N_i} w_{ij}x_{ij}^k - W\right), & \text{否则} \end{cases}$$

其中 α_0 是大的正惩罚值。因此评价函数可以表示为下式：

$$\text{eval}\ (v_k) = \frac{1}{f_k(v_k)+p_k},\quad k = 1,2,\cdots,pop_size$$

至于选择，将最优性(elitist)方法嵌入轮盘赌(roulette wheel)选择来增强其在下一代中保持最优染色体的能力并克服采样中的随机误差。在带有最优性方法的选择过程中，如果上一代的最优个体没有被轮盘赌在下一代中得到复制，则从下一代中随机去掉一个染色体并将上一代中的最优染色体加入到下一代中。

试验结果 在 Gen 等人针对该问题的遗传算法中，所有测试问题均随机产生[220]。产生 c_{ij} 和 w_{ij} 值时使用了均匀分布的随机数，以确保同一类中 c_{ij} 和 w_{ij} 的值不相同。下面的数值例子有 8 类物品，每类中有不同项目的多选择背包问题。

$$\begin{aligned} \min\quad & 3x_{11} + 4x_{12} + 5x_{13} + 4x_{14} + 8x_{15} + 5x_{16} + 4x_{17} + 4x_{18} \\ & + 7x_{21} + 9x_{22} + 8x_{23} + 4x_{24} + 5x_{25} \\ & + 5x_{31} + 6x_{32} + 9x_{33} + 8x_{34} + 4x_{35} + 6x_{36} \\ & + 2x_{41} + 8x_{42} + 7x_{43} + 5x_{44} + 4x_{45} + 3x_{46} + 7x_{47} + 8x_{48} \\ & + 8x_{51} + 5x_{52} + 7x_{53} + 9x_{54} + 7x_{55} + 4x_{56} + 7x_{57} \\ & + 6x_{61} + 8x_{62} + 7x_{63} + 9x_{64} + 5x_{65} \\ & + 6x_{71} + 9x_{72} + 7x_{73} + 4x_{74} + 3x_{75} + 7x_{76} + 9x_{77} + 8x_{78} \end{aligned}$$

$$+9x_{81}+2x_{82}+5x_{83}+5x_{84}+8x_{85}+6x_{86}$$

$$\text{s.t.}\quad 6x_{11}+8x_{12}+5x_{13}+4x_{14}+9x_{15}+5x_{16}+9x_{17}+3x_{18}$$
$$+6x_{21}+5x_{22}+3x_{23}+5x_{24}+9x_{25}$$
$$+8x_{31}+6x_{32}+4x_{33}+8x_{34}+4x_{35}+3x_{36}$$
$$+9x_{41}+4x_{42}+8x_{43}+9x_{44}+4x_{45}+9x_{46}+4x_{47}+5x_{48}$$
$$+3x_{51}+7x_{52}+4x_{53}+2x_{54}+3x_{55}+8x_{56}+5x_{57}$$
$$+8x_{61}+3x_{62}+2x_{63}+8x_{64}+6x_{65}$$
$$+5x_{71}+4x_{72}+3x_{73}+8x_{74}+9x_{75}+2x_{76}+4x_{77}+4x_{78}$$
$$+2x_{81}+9x_{82}+2x_{83}+6x_{84}+4x_{85}+3x_{86}\leqslant 40$$

$$x_{11}+x_{12}+x_{13}+x_{14}+x_{15}+x_{16}+x_{17}+x_{18}=1$$
$$x_{21}+x_{22}+x_{23}+x_{24}+x_{25}=1$$
$$x_{31}+x_{32}+x_{33}+x_{34}+x_{35}+x_{36}=1$$
$$x_{41}+x_{42}+x_{43}+x_{44}+x_{45}+x_{46}+x_{47}+x_{48}=1$$
$$x_{51}+x_{52}+x_{53}+x_{54}+x_{55}+x_{56}+x_{57}=1$$
$$x_{61}+x_{62}+x_{63}+x_{64}+x_{65}=1$$
$$x_{71}+x_{72}+x_{73}+x_{74}+x_{75}+x_{76}+x_{77}+x_{78}=1$$
$$x_{81}+x_{82}+x_{83}+x_{84}+x_{85}+x_{86}=1$$

该问题的最优值是34，3个通过枚举方法找到的最优解为

$$x_{11}=x_{24}=x_{35}=x_{45}=x_{56}=x_{63}=x_{75}=x_{83}=1$$
$$x_{18}=x_{24}=x_{35}=x_{45}=x_{52}=x_{65}=x_{75}=x_{83}=1$$
$$x_{18}=x_{24}=x_{35}=x_{45}=x_{56}=x_{65}=x_{74}=x_{83}=1$$

该问题实例的遗传系统环境的设置如下：种群规模 40，最大迭代次数 100，杂交率 0.2，变异率 0.1。采用不同的随机数种子进行了 40 次运算，以概率 0.675 找到最优解。

对各种问题规模、遗传算法参数设置、执行时间以及获得最优解概率的数值试验的比较见表 2.6。

表 2.6 例子的计算结果

问题	问题规模			遗传算法参数设置			时间/s	频率
	总项目	总类数	种群规模	最大代数	p_c	p_m		
1	10	3	20	20	0.2	0.1	0.04	1
2	15	4	20	20	0.2	0.1	0.04	1
3	31	6	40	100	0.2	0.1	0.07	0.87
4	53	8	40	100	0.2	0.1	0.15	0.67

2.4.2 多约束背包问题

问题描述 多约束背包问题(multiconstrained knapsack problem)是带有一组约束(比如重量、尺寸、可靠性等)的背包问题。该问题也称为多背包问题(multiple-knapsack problem)或多维背包问题(multidimensional knapsack problem)[441]。问题的描述如下：存在承重分别为 W_1, W_2, $\cdots$, W_m 的 m 个背包和 n 个物品，每个物品的费用为 c_j，$j=1,2,\cdots,n$。与简单背包问题中物品重量恒定所不同的是，如果第 j 个物品放人第 i 个容量为 W_i 的箱子，则它在第 i 个约束中的权重为 w_{ij}。我们希望将尽量多的物品放入背包，在确保背包承重满足的前提之下最大化总费用。带有3个约束的问题描述如图2.13所示。

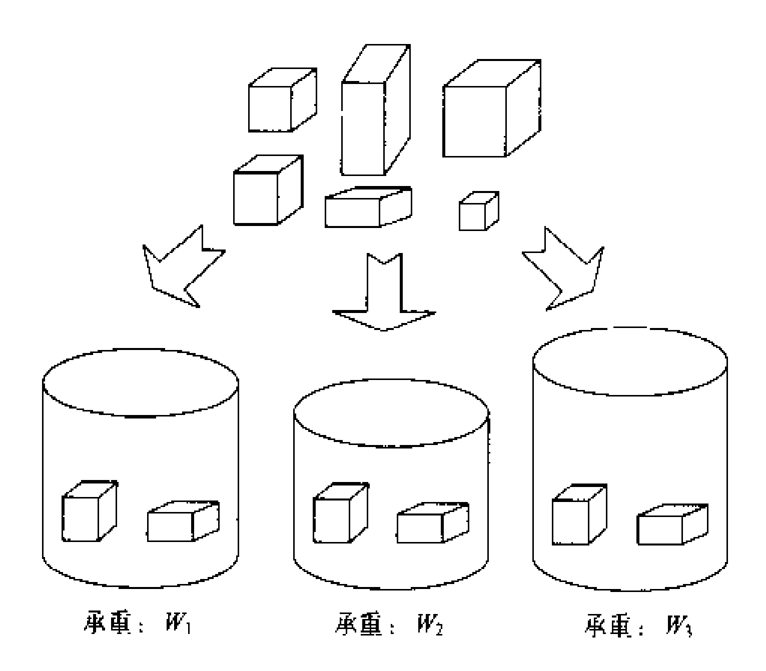


图2.13 多约束背包问题

多约束背包问题的数学描述如下：

$$\max \quad \sum_{j=1}^{n} c_j x_j \tag{2.22}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_{ij} x_j \leqslant W_i, \quad i = 1,2,\cdots,m \tag{2.23}$$

$$x_j \in \{0,1\}, \quad \forall j^{①}$$

任意满足上面二进制约束的向量或字符串 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$ 都是一个解。如果 $\boldsymbol{x}$ 满足约束(2.23)，则其为可行解。多约束背包问题是0-1线性规划的一个特例，在资源

① 译者注：x_j 的含义是：如果第 j 个物品被放人某个箱子，则 $x_j=1$，否则 $x_j=0$。

配置和资本预算这些领域中有很多应用。它也可以作为在分布式计算机系统中分配处理器和数据库模型的子问题。文献中已经针对这个 NP 难问题提出了各种算法[155,167,212]。

遗传算法求解 与其他组合优化问题类似，遗传算法已经应用于多约束背包问题。Khuri，Bäck 和 Heitkötter 直接采用二进制字符串(binary string)编码来表示问题可能的解[360]。如果字符串的第 j 位的值为 1(即 $x_j=1$)，第 j 个物品放入所有箱子，反之则不放入该物品。这样进行编码会产生不可行解。他们的方法是允许不可行解繁殖的后代加入种群，而不是将其抛弃或者忽略搜索空间的不可行区域。他们认为，如果字符串与可行区域越远，则应对其施加更多的惩罚项。每个放入箱子的物品会带来利润 $p_j(j=1,2,\cdots,n)$，问题是最大化总利润，即最大化下述适应值函数：

$$f(x) = \sum_{j=1}^{n} p_j x_j - s\max\{p_j\}$$

其中 $s = \left|\left\{i \middle| \sum_{j=1}^{n} w_{ij}x_j > W_i\right\}\right|$。换句话说，$s(0 \leqslant s \leqslant m)$表明溢出的箱子的数量。适应值函数采用了分级的惩罚项 $\max\{p_j\}$。该项削弱不可行字符串适应值的次数与该字符串产生的溢出的箱子的数量相同。他们采用上述适应值函数和并测试了若干问题[48]。他们使用了遗传算法软件包 GENEsYs[25](可以从 lupti. informatik. uni-dortmund. de ftp 站点的/pub/GA/src 目录匿名下载 GENEsYsl. o. tar. Z 获取该软件包)来解决多约束背包问题。

最近 Raidl 开发了改进遗传算法来处理多约束背包问题[522]。该算法同样采用了位字符串，但使用线性规划(LP)来产生问题的可行解。在得到基于线性规划松弛多约束背包问题的解 $x^{LP}=(x_1^{LP}, x_2^{LP}, \cdots, x_n^{LP})$①后，算法如下进行：

初始化 x 的过程

```
begin
    x ← 0
    P ← (1,2,…,n)的任意排列;
    for j ← 1 to n do
        if x^LP_P[j] > R_j then
            x_P[j] ← 1;
            if 某个 W_i 约束被违背 then
                x_P[j] ← 0;
            end
        end
```

① 译者注：x^{LP}是原问题去掉二进制约束松弛以后得到的[0,1]区间上的解。

```
        end
    end
```

将所有的变量 x_j 初始化为0。然后采用下标1到 n 的随机排列来按照随机的顺序处理所有变量 x_j。在余下的循环中，每个变量 $x_{P[j]}$ 以LP解值 $x_{P[j]}^{LP}$ 为概率取为1。该过程中采用了伪随机数 $R_j(0\leqslant R_j\leqslant 1)$。如果将变量 $x_{P[j]}$ 取为1违反了任何约束，则 $x_{P[j]}$ 被重设为0。于是仅产生可行解。由于随机排列的作用，该算法产生不同的初始解，从而维持种群多样性。

标准均匀杂交和按位进行的变异可能产生不可行解。该算法不采用惩罚的方法，而使用了一种修补算子。该算子如下进行：

修补 x 的过程

```
begin
    P ← (1,2,…,n)的任意排列同时满足 x^LP_P[j] ≤ x^LP_P[j+1]  (j=1,…,n-1)；①
    for j ← 1 to n do
        if x_P[j] =1 而且有任何 W_i 约束被违背 then
            x_P[j] ← 0；
        end
    end
end
```

该算子在每个不可行解进行适应值评价之前执行。同样首先采用了下标1到 n 的随机排列来确定检查变量 x_j 的顺序。但还需要根据LP解值 x_j^{LP} 上升的顺序进行排序。也就是仅对LP解值相同的下标进行随机排列。由于具有较大 x_j^{LP} 值的变量在设为1后更有价值，因此从具有较小 x_j^{LP} 值的变量开始处理。在循环过程中，所有被设为1的变量 $x_{P[j]}$ 被依次检查，如果违背了约束 W_i，则该变量被设为0。所以在最坏情况下，所有变量设为0，x 确保是可行解。

进一步采用了一种局部改进方法在确保解可行之后对解进行改进。局部改进方法如下所示：

局部改进 x 的过程

```
begin
    P ← (1,2,…,n)的任意排列同时满足 x^LP_P[j] ≥ x^LP_P[j+1]  (j=1,…,n-1)；②
```

① 译者注：原问题松弛后求得的解中越接近0的元素在非松弛环境中违反约束的可能性越大。初始化过程可能让这种元素的值为1。在修改过程中应该首先考虑将这种元素的值修补为0。另一方面，松弛问题的解中越接近1的元素对目标函数的贡献越大，因此应该推迟将其修补为0的次序。

② 译者注：松弛问题的解中越接近1的元素对目标函数的贡献越大。初始化过程和修补过程可能让这种元素的值为0。在局部改进过程中应该首先考虑将这种元素的值改为1并检查是否违反约束。

```
        for j ← 1 to n do
            if x_P[j] = 0 then
                x_P[j] ← 1;
                if 任何 W_i 约束被违背 then
                    x_P[j] ← 0;
                end
            end
        end
    end
```

与修补过程不同，变量 x_j 按照 LP 解值 x_j^{LP} 下降的顺序来处理。与初始化过程类似，如果不违背约束，则 $x_{P[j]}=1$。因此解可能在维持可行性的基础之上加以改进。改进遗传算法的主要优势是很快的收敛速度，特别在大规模问题时更是如此。

2.5 最小生成树问题

最小生成树问题(minimum spanning tree problem)在组合优化中具有悠久的历史。Boruvka 于 1926 年首次提出该问题，目的是寻找电力线网络最经济的布局[262]。此后最小生成树问题被广泛应用于许多组合优化问题中，比如运输问题、通信网络设计，分布式系统等[357]。

考虑连通图 $G=(V,E)$，其中 $V=\{v_1,v_2,\cdots,v_n\}$ 是顶点(vertices)的有限集合，$E=\{e_1,e_2,\cdots,e_m\}$ 是边(edges)的有限集合。边将顶点之间连接起来。每个边有一个正实数权重 $W=\{w_1,w_2,\cdots,w_m\}$ 用来表示距离或费用。最小生成树问题就是寻找图 G 中连接所有顶点的具有最小权重的子图。

设 $\boldsymbol{x}$ 为一个二进制决策变量。如果边 e_i 被选中，则 $x_i=1$，否则 $x_i=0$。设 T 代表图 G 的所有生成树的集合。最小生成树问题数学上的表示为

$$\min\left\{z(\boldsymbol{x})=\sum_{i=1}^{m}w_ix_i \mid \boldsymbol{x}\in T\right\} \tag{2.24}$$

对最小生成树问题的算法进行了大量的研究，产生了多种快速算法。这些算法可以在近似线性的时间复杂度内进行求解，时间复杂度与边的数量或顶点数量相关[164,376,515,579]。

许多研究工作表明，最小生成树结构是通信网络设计的最优拓扑[357]。生成树在大多数网络设计和分析问题中扮演重要角色。然而，实际的网络优化问题通常需要满足附加的约束。因此形成了约束最小生成树问题(constrained minimum spanning tree problem)[160]，其中的一些列表显示于表 2.7。

表 2.7　约束最小生成树问题的例子

作　　者	问　　题
Bertsimas (1990)[59]	概率最小生成树问题
Fernandes 和 Gouveia (1998)[186]	叶约束最小生成树问题
Ishii, Shiode 和 Nishida (1981)[320]	随机最小生成树问题
Kershenbaum (1974)[356]	容量限制的最小生成树问题
Narula 和 Ho (1980)[482]	度约束的最小生成树问题
Xu (1984)[675]	二次最小生成树问题
Zhou and Gen (1996)[696]	多判据最小生成树问题

大多数约束最小生成树问题是 NP 难的，目前不存在多项式时间解。鉴于这种问题的复杂性，一些研究人员采用遗传算法来进行处理(参阅表 2.8)。

表 2.8　遗传算法用于约束最小生成树问题

作　　者	问　　题
Abuali, Wainwright 和 Schoenfeld (1995)[8]	概率最小生成树问题
Palmer 和 Kershenbaum (1995)[509]	概率最小生成树问题
Zhou and Gen (1997)[699,701]	度约束的最小生成树问题
Zhou and Gen (1998)[697]	度约束的最小生成树问题
Zhou and Gen (1997)[699]	容量限制的最小生成树问题
Zhou and Gen (1998)[702]	二次最小生成树问题
Zhou and Gen (1998)[703]	多判据最小生成树问题
Zhou and Gen (1998)[697]	叶约束最小生成树问题

2.5.1　二次最小生成树问题

二次最小生成树问题(quadratic minimum spanning tree problem)考虑两种类型的费用：直接费用和交互费用。直接费用(direct cost)是与每边相关的费用。交互费用(interactive cost)的概念由 Xu[675]引入，用来描述同时将一对边选入树时产生的费用。由于考虑了交互费用，式(2.24)中目标函数不再线性。设 c_{ik} 表示由一对边(e_i,e_k)引起的交互费用。问题可以表述为下面的形式：

$$\min\left\{z(x)=\sum_{i=1}^{m}\sum_{k=1,k\neq i}^{m}c_{ik}x_ix_k+\sum_{i=1}^{m}w_ix_i \mid x\in T\right\} \tag{2.25}$$

带有上述二次目标函数的最小生成树问题就是二次最小生成树问题[675]。

启发式算法(heuristic algorithms)　二次最小生成树问题目前没有有效的多项式时间算法，仅存在 Xu 给出的两种启发式算法[675]。

启发式算法 H1(平均贡献方法)　如果单独考虑边 e_k 并且将所有与下标 k 相关的项

分离出来，式(2.25)中的目标可以重写为

$$z(\boldsymbol{x}) = \left[w_k + \sum_{j \neq k}(c_{jk} + c_{kj})x_j\right]x_k + z_2(\boldsymbol{x})$$

其中项 $z_2(\boldsymbol{x})$不再包含 x_k。括号内存在$(m-1)$项的求和，这些项中除了$(n-1)$项以外都一定是 0[①]。设 e_k 是求得的树中的一条边。e_k 对于目标函数的平均贡献可以根据下式估计：

$$p_k = w_k + \frac{n-1}{m-1}\sum_{j \neq k}(c_{jk} + c_{kj}), \quad 1 \leqslant k \leqslant m \tag{2.26}$$

在评价完所有边 $e_k(k=1,2,\cdots,m)$的平均贡献后，q - MST 问题的解可以通过求解下面的 MST 问题得到[②]：

$$P = \min\left\{\sum_{k=1}^{m} p_k x_k \mid \boldsymbol{x} \in T\right\} \tag{2.27}$$

启发式算法 H2(顺序固定方法)　在启发式算法 H1 中，所有边的平均贡献是独立估计的。假设固定树中的某些边将影响对其余边贡献的估计。其余边的平均贡献在固定其他边之前被重新评估。这种方式顺序确定生成树中的边。

设 U 是已经被选入树的边的集合，S 是已经被排除的边的集合(如果加入这些边，将形成回路)，F 是余下的可能被选入树的边的集合。每次迭代中边 e_k 的平均贡献可以顺序估计如下：

$$q_k = \hat{w}_k + \frac{n_1}{m_1}\sum_{j \in F, j \neq k}(c_{jk} + c_{kj}) \tag{2.28}$$

其中 $m_1 = |F| - 1$，$n_1 = n - 1 - |U|$[③]。这意味着$|F|-1$ 项中除了 $n-1-|U|$项以外都成为零贡献。另外，

$$\hat{w}_k = w_k + \sum_{j \in U}(c_{jk} + c_{kj})$$

启发式算法可以用下面的过程来说明：

启发式算法 H2 的过程

第 1 步：设 $U \leftarrow \{\varnothing\}$，$F \leftarrow \{1,2,\cdots,m\}$，$n_1 \leftarrow n-1$，$m_1 \leftarrow m-1$。

第 2 步：对于所有 $k \in F$ 计算 q_k。

第 3 步：选择 e_l 使得 $q_l = \min\{q_k \mid k \in F\}$。设 $x_l \leftarrow 1$，$U \leftarrow U \cup \{l\}$，$F \leftarrow F \backslash \{l\}$，$n_1 \leftarrow n_1 - 1$，$m_1 \leftarrow m_1 - 1$。如果 $n_1 = 0$，停止。

---

① 译者注：有 n 个顶点的图的树中边的数量为 $n-1$。如果边 e_k 被选入树，则与其他构成树的边的交互费用有 $n-1$ 项。

② 译者注：MST 是 minimum spanning tree 的缩写，q - MST 是 quadratic minimum spanning tree 的缩写。

③ 译者注：m_1 是余下可能被选入树的边的集合中除去 e_k 边以外边的数量，n_1 是生成树尚需的边的数量。

第4步：如果对于 F 中任意 k，添加 e_k 会与 U 中的边形成回路，设 $x_k \leftarrow 0$，$F \leftarrow F \setminus \{k\}$。返回第2步。

遗传算法实现 Zhou 和 Gen 提出了用于解决二次最小生成树的遗传算法[702]。由于 Prüfer 数(Prüfer number)可以代表所有的树，他们采用 Prüfer 数来表示问题所有可能的解。图 2.14 说明了这种编码。详细描述请读者参考 Gen 和 Cheng 的文章[219]。

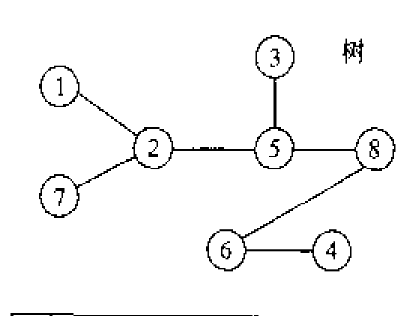


图 2.14 树及其 Prüfer 数

杂交和变异 由于在任何杂交或变异操作之后 Prüfer 数编码总会表示一棵树，故可以采用均匀杂交算子。变异则采用从 1 到 n 范围内的随机扰动来进行。

评价和选择 评价包括下面两个步骤：

1. 将染色体转换为树。这个过程就是将 Prüfer 数转换为树的过程，树用边的集合来表示。

2. 计算树的总费用。其中每个染色体的适应值可以根据式(2.25)目标函数直接计算。

采用了$(\mu+\lambda)$选择方法[30]。该方法从 μ 个父代和 λ 个子代中选出 μ 个最好的染色体作为后代。如果不能选出 μ 个不同的染色体，空的种群池由新产生的个体填充。

数值例子 测试问题的规模从 10 到 50 个顶点不等[702]。所有图都是包含 n 个顶点和 $m=n(n-1)/2$ 个边的完全图。每个二次最小生成树问题的交互费用矩阵的对角线元素都是从(0,100]范围内随机选出的整数。交互费用矩阵的非对角线元素都是从(0,20]范围内随机选出的整数。

遗传算法的参数设置如下：种群规模 $pop_size=300$，杂交率 $p_c=0.2$，变异率 $p_m=0.2$，最大遗传代数 $max_gen=500$，运行次数为 20。

表 2.9 对 Xu 的启发式算法和 Zhou 的遗传算法进行了比较。显然遗传算法得到了好得多的结果。使用遗传算法可以比启发式算法得到的最好解平均改善 9.6%(最多改善 12.83%)。

表 2.9 启发式算法和遗传算法的比较

问题规模(顶点数)	H1	H2	GA	改进(%)
10	925	921*	834	9.45
20	3554*	3749	3233	9.03
30	8500*	9402	7742	8.92
40	15 539	14 805*	14 275	3.58
50	25 182	23 487*	22 980	2.16

注：H1 和 H2 是 Xu 提出的启发式算法；GA 是 Zhou 和 Gen 提出的遗传算法。星号显示启发式算法得到的最好解。

改进 $=(H_{best}-GA)/H_{best}\times 100\%$，其中 H_{best} 是启发式算法得到的最好解。

2.5.2 度约束的最小生成树问题

在最小生成树问题中，如果假设存在每个顶点的度约束，即对于每个顶点 v_j，其度值 d_j 不能超过给定的数值 b_j，则连接到每个顶点的边的数量受到了限制①。这时问题成为*度约束的最小生成树问题*(degree-constrained minimum spanning tree problem)，可以表示为下面的形式：

$$\min\left\{z(\boldsymbol{x})=\sum_{i=1}^{m}w_i x_i \mid d_j \leqslant b_j, j \in V, \boldsymbol{x} \in T\right\} \tag{2.29}$$

一种根据直觉方法处理该问题的思路是采用启发式算法：(1)搜索不带度约束的最小生成树以及(2)修改找到的最小生成树使其满足度约束。但是随着问题规模的逐渐增大，有效率地操作变得比较困难。Narula 和 Ho[482]，Savelsbergh[562] 和 Volgenant[644] 都提出了解决这个问题的启发式算法。Zhou 和 Gen 提出了处理这类问题的遗传算法[219]。最近，Zhou 和 Gen 又开发了一种新的基于树的度值的编码方式来求解度约束最小生成树问题[697]。

遗传算法实现

基于度的排列(degree-based permutation)　对于度约束的最小生成树问题，需要将两个因素编码入染色体：(1)顶点之间的连接关系，(2)每个顶点的度值。因此，直接的思路是采用二维结构来对带有度约束的生成树进行编码：其中一维对顶点之间的连接关系进行编码②，另一维对每个顶点的度值进行编码③。因此需要一个 $2\times n$ 矩阵来表示 n 个顶点树的染色体。顶点维中基因不重复地在整数 1 到 n 中进行取值，度维中基因在整数 1 到 b 中进行取值，b 是所有顶点的度约束。这种表示被 Zhou 和 Gen 称作基于度的排列[697]。

对于无向树，我们可以将任何顶点作为根顶点。所有其他顶点可以看成分级连接到根顶点上。对于一个给定的顶点(当前顶点)，与其相连的上一级顶点称作前任顶点(predecessor vertex)，与其相连的下一级顶点称作继任顶点(successor vertex)。显然根顶点没有前任顶点，叶顶点没有继任顶点。基于这种观察，树的基于度的排列可以根据下述过程进行编码：

基于度的排列的编码过程

第 1 步：在一棵树 T 中任意选择一个顶点作为根顶点，将其顶点编号作为染色体顶点维的排列中第 1 个数，将其度值作为染色体度维中第 1 个数。将该顶点作为当前

① 译者注：无向图中某一顶点的度就是不包括自回路在内的所有连接到该顶点的边的数量。

② 译者注：称作顶点维。

③ 译者注：称作度维。

顶点。

第2步：从树的左分支到右分支搜索当前顶点的继任顶点。如果存在继任顶点，将其顶点编号放入顶点维，度值放入度维(这里我们采用向右添加的方式生成排列)，然后进入第3步。如果不存在当前顶点的继任顶点，将当前顶点的前任顶点作为当前顶点，返回第2步。

第3步：如果继任顶点不是叶顶点，将继任顶点作为当前顶点，返回第2步。如果继任顶点是叶顶点，删除它，进入第4步。

第4步：如果所有顶点都被检查过，停止。否则返回第2步。

图2.15说明了基于度的排列。

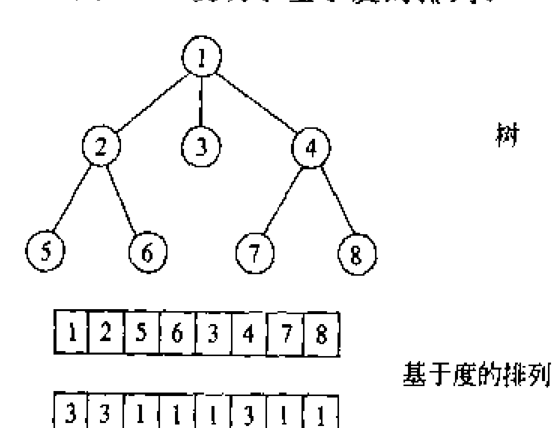


图2.15 树及其基于度的排列

从基于度的排列中获得树是比较容易的。设 P_1 表示给定个体的顶点维，P_2 表示给定个体的度维。解码过程如下所示：

基于度的排列的解码过程

第1步：设 $k\leftarrow1, j\leftarrow2$。

第2步：将 $P_1(k)$代表的顶点称作 v_r，将$P_1(j)$代表的顶点称作 v_s，连接 v_r 和 v_s。

第3步：$P_2(j) \leftarrow P_2(j)-1$, $P_2(k) \leftarrow P_2(k)-1$。

第4步：如果 $P_2(j)\geq1$，则 $k \leftarrow j$, $j \leftarrow j+1$。

第5步：如果 $k<1$，停止。

第6步：如果 $P_2(k)\geq1$，将 $P_1(k)$代表的顶点称作 v_r，$P_2(k) \leftarrow P_2(k)-1$；否则 $k \leftarrow k-1$，返回第5步。[①]

第7步：如果 $P_2(j)\geq1$，将 $P_1(j)$代表的顶点称作 v_s，$P_2(j)=P_2(j)-1$；否则 $j \leftarrow j+1$，返回第7步。[②]

第8步：连接 v_r 和 v_s，返回第4步。[③]

显然，任何生成树都可以用上述方法进行编码表示，同时任何编码都代表了一棵树。然而，编码和生成树的关系不是无向图上的1-1映射，不同的编码可能代表同一生成树。编码过程在某种意义上保持了位置性，即基因型(编码)上的小变化在表现型(生成树)上也是小变化。

---

① 译者注：如果当前顶点的度已经给完全使用(当前顶点在树中没有与其他边相连)，则返回当前顶点的前任顶点。

② 译者注：如果当前顶点的度已经给完全使用(当前顶点在树中没有与其他边相连)，则进入当前顶点的继任顶点。

③ 译者注：此时 k 所代表的顶点总是 j 所代表的顶点的前任顶点。此时的 $P_2(k)$、$P_2(j)$分别代表树中顶点 $P_1(k)$和顶点 $P_1(j)$尚未由解码过程产生的边的数量。

初始化种群 为了保持顶点的度约束和顶点之间的连接性,染色体度维中的基因需要满足下面的条件:对于 n 顶点的树,任何顶点的度值都不小于 1,所有顶点度值之和为 $2(n-1)$。设 d_{rest} 代表所有度值尚未在染色体的度维中指定的顶点的度值之和的下界,d_{used} 代表所有度值已经在染色体的度维中指定的顶点的度值之和。于是当前顶点在度维中度值的取值范围是:不小于 1 也不大于给定的度值。当前顶点与余下顶点的度值之和应该满足:不小于 d_{rest},也不大于 $2(n-1)-d_{used}$。

顶点的次序杂交(order crossover) 次序杂交由 Davis[145] 提出。为了避免顶点间的非法连接,杂交仅在顶点维中进行,度维保持不变。杂交过程用图 2.16 表示。杂交在遗传各代之间将极大地改变树的结构。这对于广度搜索整个解空间是有益的。

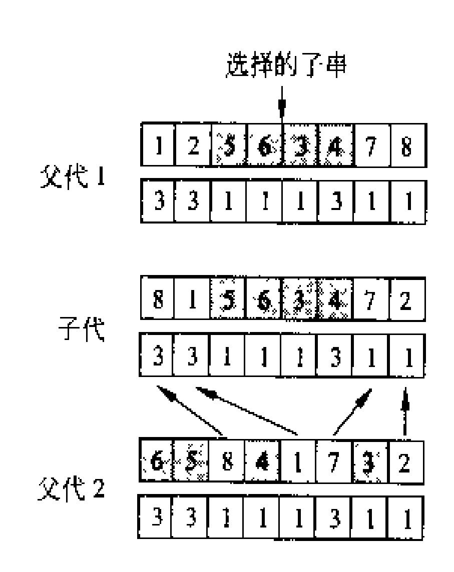


图 2.16 顶点的次序杂交

顶点的交换变异(swap mutation) 交换变异随机选择两个基因(顶点),然后交换他们。变异过程用图 2.17 表示。

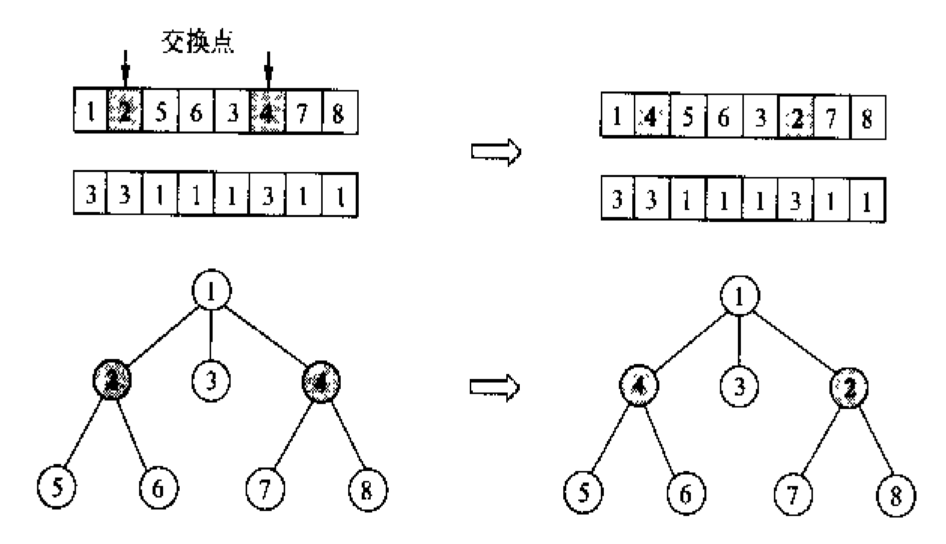


图 2.17 顶点的交换变异

插入变异(insertion mutation) 插入变异从顶点维中随机选出一段基因(树的分支),然后将其插入一个随机选取的基因(顶点)后面。在移开基因和插入基因过程中需要修改度维中对应的基因(度),参见图 2.18。这种算子可以在代与代之间维持良好的继承性。这种变异的结果是进化过程的深度搜索。

数值例子 表 2.10 表示的是一个简单的 9 顶点完全图,其中所有顶点的度约束都是 3。该问题由 Savelsbergh 和 Volgenant 给出,他们采用称作边交换(edge exchange)的启发式算法对其进行优化,得到的最优解是 2256[562]。

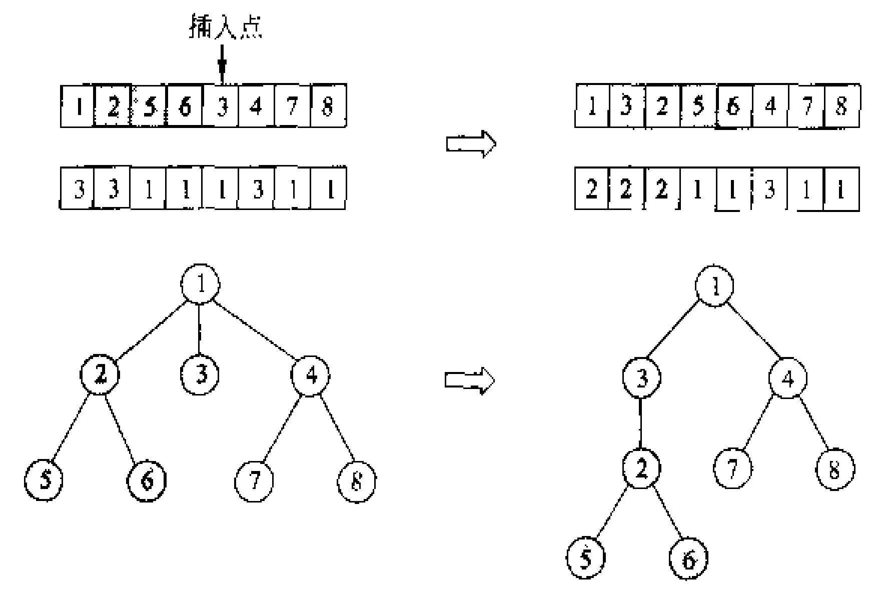


图 2.18　插入变异

表 2.10　9 顶点问题的边权重

i	1	2	3	4	5	6	7	8	9
1	—	224	224	361	671	300	539	800	943
2		—	200	200	447	283	400	728	762
3			—	400	566	447	600	922	949
4				—	400	200	200	539	583
5					—	600	447	781	510
6						—	283	500	707
7							—	361	424
8								—	500
9									—

由 Zhou 和 Gen 提出的基于度排列编码方式的遗传算法参数如下：种群规模 $pop_size=100$，杂交率为 0.4，用于交换的变异率为 0.2，用于插入的变异率为 0.6，最大遗传代数 $max_gen=500$。

采用遗传算法可以轻松获得最优解 2256。表 2.11 表示了 5 次随机产生实例的计算结果。每边的权重随机产生，在[10,100]区间上均匀分布。每个实例的下界采用 Prim 算法[515]松弛度约束获得。采用遗传算法得到的结果在实例下界的 6%以内。

表 2.11　采用遗传算法求得的结果和问题下界的比较[a]

问题规模(顶点)	LBs		dpGAs	
	min_val	度	min_val	%
10	117	5(2)	123	5.13
20	233	4(2)	237	1.72
30	316	4(4)	321	1.58
40	419	7(3)	428	2.15

a：LBs 代表问题的下界；dpGAs 代表采用基于度的排列编码的遗传算法；min_val 代表最小值；%代表遗传算法得到的最小值与问题下界相差的比例。

2.5.3 双目标最小生成树问题

现实世界中常常有这样的情况，由于图的每个边定义了多个属性，在决策最小生成树问题时需要同时考虑多个目标。比如说，当设计通信系统的布局时，除了城市或终端之间的连接费用之外，其他因素也很重要，同样需要考虑。这些因素包括通信或建设的时间、建设的复杂性，甚至还有可靠性。在这种情况下，带有多个目标的最小生成树问题代表了更为实际的问题[598]。

让我们从式(2.24)表示的最小生成树问题出发。如果每边有两个与其相关的正实数，代表着定义在该边上的两个属性，并用 $w_i=(w_{1i},w_{2i})$ $(i=1,2,\cdots,m)$。于是**双目标最小生成树问题**(bicriteria minimum spanning tree problem)就可表示为

$$\left.\begin{aligned} \min \quad & z_1(x)=\sum_{i=1}^{m} w_{1i}x_i \\ \min \quad & z_2(x)=\sum_{i=1}^{m} w_{2i}x_i \\ \text{s.t.} \quad & x\in T \end{aligned}\right\} \tag{2.30}$$

枚举法 通常我们无法获得该问题的最优解，原因在于两个目标在实际中常常相互冲突。这种问题的解是一组 Pareto 最优解[97]。一种可能采取的，同时也是处理多目标优化问题经常使用的方法就是将两个目标按比例合成为一个目标。但这样做仅能获得一个 Pareto 最优解，更不要说将两个目标恰当地转换问题一个目标的难度了。然而在实践中，决策者通常对一个(Pareto 最优)点更感兴趣。这种决策取决于他对不同目标的偏好。获得(所有)其他的 Pareto 最优解是有用的。Zhou 和 Gen 针对双目标最小生成树问题提出了一种枚举法，通过枚举所有 Pareto 最优解来验证他们提出的遗传算法的效果和效率。

模拟获得最小生成树过程中的边生长过程可以枚举所有的生成树，从而找到问题的所有非支配解。在边生长的过程中，我们无法知道哪个边具有最小的权重。但我们可以知道如果将一条边加入一棵特定的子树，这棵子树是否被支配。因此，一旦某边被加入一棵子树，非支配子树集合就进行更新。最后，当所有子树有 $n-1$ 条边时，我们可以获得所有非支配解的集合。

在枚举所有 Pareto 最优解过程中需要考虑两个重要因素：

1. 对于非支配子树集合中的所有子树，每一步检查所有相邻的边，但仅将一条边加入子树。

2. 由于每棵子树可生成多于一棵的非支配子树，因此从不同子树中生成的带有相同边数量的所有非支配子树需要重新检查其 Pareto 最优性，并更新非支配子树集合。

设 $T_r^{(t)}$ 是包含 t 条边的第 r 棵子树。$r\in I^{(t)}$，$I^{(t)}$ 代表包含 t 条边的子树的下标集合。

$W_k(T_r^{(t)})$是子树 $T_r^{(t)}$ 在第k个目标上的总权重。$E(T_r^{(t)})$是所有与子树 $T_r^{(t)}$ 相邻的边的集合。

枚举过程

第1步：设 $t \leftarrow 0, T_0^{(0)} \leftarrow \varnothing$，$I^{(0)} \leftarrow \varnothing$。$W_k(T_0^{(0)}) \leftarrow 0$，$k=1,2$。

第2步：从V中选择任意一个顶点，称为v_1，设$E(T_0^{(0)})$是包含所有与该顶点相邻的边的集合。设 $r \leftarrow 1$，$t \leftarrow t+1$。到第6步。

第3步：设 $r \leftarrow 1, t \leftarrow t+1$。如果 $t \geqslant n-2$，终止过程。否则继续。

第4步：如果 $r \notin I^{(t-1)}$，返回第3步。否则继续。

第5步：如果 $E(T_r^{(t-1)})$是空集，则 $r \leftarrow r+1$，返回第4步。否则继续。

第6步：从集合 $E(T_r^{(t-1)})$中随机选出一条边e_l。更新边的集合 $E(T_r^{(t-1)}) \leftarrow E(T_r^{(t-1)}) \backslash \{e_l\}$。

第7步：如果边$\{e_l\}$与 $T_r^{(t-1)}$ 中所有其他边构成回路，返回到第5步。否则继续。

第8步：建立临时子树 $T^{(t)} \leftarrow T_r^{(t-1)} \cup \{e_l\}$，并用权重 $W_k(T_r^{(t)}) \leftarrow W_k(T_r^{(t-1)}) + w_{lk}$ $(k=1,2)$来计算该子树的两个目标函数值。

第9步：将子树 $T^{(t)}$ 与 $T_s^{(t)}$ $(s \in I^{(t)})$中所有子树在 Pareto 最优性意义上进行比较。如果该临时子树是非支配子树，更新非支配子树集($T_s^{(t)}$，$s \in I^{(t)}$)和下标集合 $I^{(t)}$，返回第5步。

当所有这些子树 $T_s^{(t)}$，$s \in I^{(t)}$都包含 $n-1$ 条边时，就得到了问题最终的 Pareto 最优解。显然该方法可以向多目标最小生成树问题扩充。

遗传算法　Zhou 和 Gen 提出了处理双目标最小生成树问题的遗传算法[703]。该算法采用 Prüfer 数对树进行编码[517]。其原因是这种编码方式能够惟一表示图中所有可能的生成树，而且采用遗传算法也容易处理这种编码。和 Prüfer 数一起采用的遗传操作是均匀杂交[603]和扰动变异。

Zhou 和 Gen 在遗传算法中针对双目标最小生成树问题采用了两种策略来评价每个个体的适应值。第一种策略是使用基于3.6节讨论的方法的适应性评价函数(adaptive evaluation function)。评价过程如下所示：

评价策略Ⅰ的过程

第1步：将所有染色体解码并根据每个目标函数计算它们的目标函数值。

第2步：根据下式确定所有个体的适应值函数 eval(T)：

$$\mathrm{eval}(T) = \sum_{k=1}^{2} \lambda_k z_k$$

其中 λ_k $(k=1,2)$是权重系数。

仅采用$(\mu+\lambda)$选择方法。

第二种策略采用3.6节讨论的非支配排序方法。整个过程可以简单描述如下：

评价策略Ⅱ的过程

第1步：从当前种群确定所有非支配个体 P_c，并为它们设置比较大的哑适应值(dummy fitness value)。

第2步：计算当前非支配解集中每个个体的小生境数(niche count)m_j：

$$m_j = \sum_{k \in P_c} \mathrm{sh}(d_{jk})$$

其中

$$\mathrm{sh}(d_{jk}) = \begin{cases} 1 - \left(\dfrac{d_{jk}}{\sigma_{\mathrm{share}}}\right)^2, & \text{如果 } d_{jk} < \sigma_{\mathrm{share}} \\ 0, & \text{否则} \end{cases}$$

d_{jk} 是当前非支配解集中个体 j 和个体 k 之间的表现型距离，σ_{share} 是两个个体成为相同小生境成员所必须的最大表现型距离。

第3步：通过将哑适应值除以每个个体的小生境数来计算其共享后的适应值。

第4步：忽略所有已经进行共享处理的个体，返回第1步，重复本过程直到整个种群都进行过共享处理为止。

用于求解双目标最小生成树问题的总伪代码如下所示：

遗传算法过程

```
begin
    t ← 0;
    初始化父代种群 P(0);
    确定非支配解集合 E(0);
    while 不满足停止条件 do
    begin
        重组 P(t)产生子代种群 C(t);
        更新 E(t);
        if 选择评价策略 I then
            用策略 I 评价 P(t)和 C(t);
            从 P(t)和 C(t)中用评价策略 I 选出 P(t+1);
        else
            用策略 II 评价 P(t)和 C(t);
            从 P(t)和 C(t)中用评价策略 II 选出 P(t+1);
        end
        t ← t+1;
    end
end
```

数值例子　遗传算法用于多目标最小生成树问题的性能采用随机生成的5个从10

到50个顶点的完全图作为数值例子来进行测试。每条边有两个权重。权重随机产生，分别在[10,100]和[10,50]上均匀分布。

遗传算法的参数如下：种群规模 $pop_size=200$，杂交率 $p_c=0.2$，变异率 $p_m=0.05$，最大遗传代数 $max_gen=500$。

表2.12说明遗传算法比Pareto最优枚举方法要有效得多。当问题规模比较小的时候，所有遗传算法得到的解都是Pareto最优解。随着问题规模的增大，大多数遗传算法得到的解是Pareto最优解。

表2.12　遗传算法和枚举方法的结果[a]

问题规模（顶点）	枚举		GA+策略Ⅰ		GA+策略Ⅱ	
	TPO	运行时间/min	%	运行时间/min	%	运行时间/min
10	56	20	100	3.02	100	4.20
20	123	56	100	4.31	100	6.50
30	262	132	90	6.18	94	9.32
40	417	336	86	8.05	91	12.12
50	635	744	80	10.96	85	15.58

a：TPO代表Pareto最优解的总数；%代表求得的解占总Pareto最优解的百分数。

用遗传算法和策略Ⅰ进行10次试验的平均结果如图2.19和图2.20所示。图中既给出了理想点，又给出了Pareto最优解。两幅图清楚地表明采用适应性评价函数后，所有Pareto最优解与理想点很接近。在图2.20表示的50个顶点的例子中，很明显适应性评价函数强制个体向理想点进化，并将注意力放到接近理想点的Pareto边界上。这种行为表明了决策者的偏好，即所有目标具有相同的重要性，所有目标与理想点越近越好。

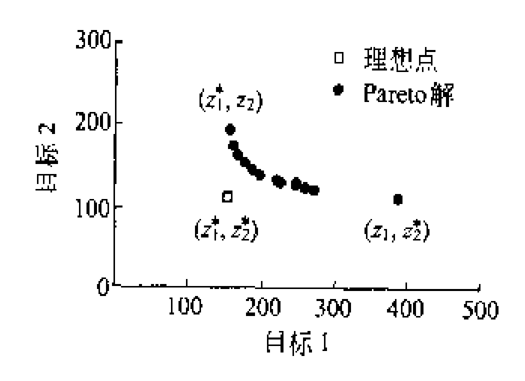


图2.19　用遗传算法和策略Ⅰ优化10顶点mc-MST

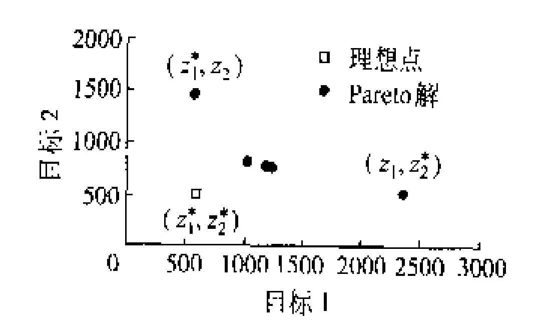


图2.20　用遗传算法和策略Ⅰ优化50顶点mc-MST

图 2.21 和图 2.22 表示采用策略Ⅱ的结果。它们表明所有解沿着 Pareto 边界(而不是特定的区域)分布。这种方法提供了更多解,允许决策者在其中进行选择。

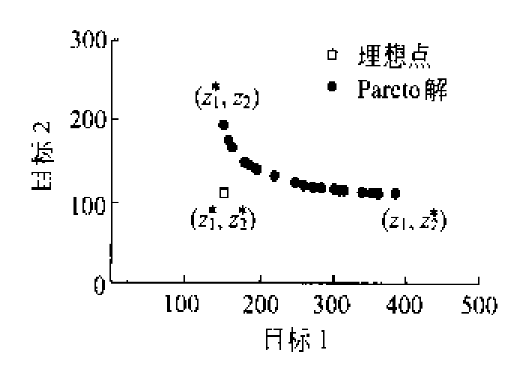


图 2.21 用遗传算法和策略Ⅱ优化 10 顶点 mc-MST

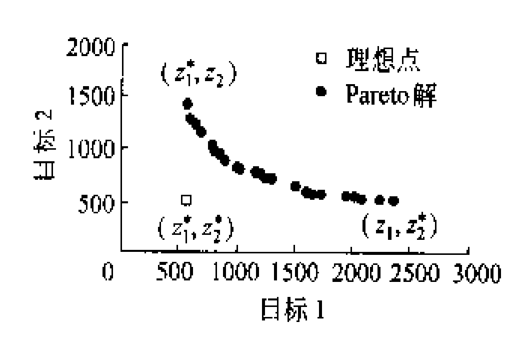


图 2.22 用遗传算法和策略Ⅱ优化 50 顶点 mc-MST

第3章 多目标优化问题

3.1 引言

最优化处理的是在一堆可能的选择中搜索对于某些目标来说是最优解的问题。如果仅考虑一个目标，就成为单目标优化问题，这种问题在过去50年中已经得到了深人的研究。如果存在的目标超过一个并需要同时处理，就成为多目标优化问题[161,588]。多目标优化问题起源于许多实际复杂系统的设计、建模和规划问题，这些系统所在的领域包括工业制造、城市运输、资本预算、森林管理、水库管理、新城市的布局和美化、能量分配等等。几乎每个重要的现实生活中的决策问题都要在考虑不同约束的同时处理若干相互冲突的目标，这些问题大大增加了问题的复杂程度。自20世纪60年代早期以来，多目标优化问题吸引了越来越多不同背景研究人员的注意力[313]。许多学者对多目标优化问题做出了重要贡献。其中 Pareto 可能是本领域公认的开创者之一[503]。感兴趣的读者可以参考文献[583]和[584]。遗传算法作为解决多目标优化问题的新方法受到了相当的关注，这就导致了一类新的研究和应用，称作遗传多目标优化。

3.2 多目标优化的基本概念

单目标优化问题通常的表述为下面的形式：

$$\max \quad z = f(\boldsymbol{x}) \tag{3.1}$$

$$\text{s.t.} \quad g_i(\boldsymbol{x}) \leqslant 0, \quad i = 1,2,\cdots,m \tag{3.2}$$

其中 $\boldsymbol{x} \in \mathbb{R}^n$ 是带有 n 个决策变量的向量，$f(\boldsymbol{x})$ 是目标函数，$g_i(\boldsymbol{x})$ 是 m 个不等式约束函数，由它们形成了可行解区域。通常在决策空间中用 S 来表示可行区域，如下所示：

$$S = \{\boldsymbol{x} \in \mathbb{R}^n \mid g_i(\boldsymbol{x}) \leqslant 0, \quad i = 1,2,\cdots,m, \boldsymbol{x} \geqslant \boldsymbol{0}\} \tag{3.3}$$

不失一般性，多目标优化(multiobjective optimizations)问题可以表述为下面的形式：

$$\max \quad \{z_1 = f_1(\boldsymbol{x}), \quad z_2 = f_2(\boldsymbol{x}), \cdots, z_q = f_q(\boldsymbol{x})\} \tag{3.4}$$

$$\text{s.t.} \quad g_i(\boldsymbol{x}) \leqslant 0, \quad i = 1,2,\cdots,m \tag{3.5}$$

有时需要在决策空间和判据空间中用图形来表示多目标优化问题。S 用来表示决策空间(decision space)中的可行区域，Z 用来表示判据空间(criterion space)的可行区域。

$$Z = \{\boldsymbol{z} \in \mathbb{R}^q \mid z_1 = f_1(\boldsymbol{x}), z_2 = f_2(\boldsymbol{x}), \cdots, z_q = f_q(\boldsymbol{x}), \boldsymbol{x} \in S\} \tag{3.6}$$

其中 $\boldsymbol{z} \in \mathbb{R}^q$ 是带有 q 个目标函数的向量。换句话说，Z 是 S 中所有点的象的集合。虽然 S

被限制为$\mathbb{R}^n$上的非负象限，但 Z 却没有限制为$\mathbb{R}^q$上的非负象限。

3.2.1 非支配解

多目标优化问题与单目标优化问题的差异非常大。在有单个目标时，人们寻找最好的解，这个解比其他所有的解都要好。在有多个目标时，由于存在目标之间的无法比较和冲突现象，不一定有在所有目标上都是最优的解。一个解可能在某个目标上是最好的，但在其他目标上是最差的。因此在有多个目标时，通常存在一系列无法简单进行相互比较的解。这种解称作*非支配解*(nondominated solutions)或 *Pareto 最优解*(Pareto optimal solutions)，它们的特点是：无法在改进任何目标函数的同时不削弱至少一个其他目标函数。对于一个给定的判据空间 Z 中的非支配解，它在决策空间中的原象点称作*有效的*(efficient)或*非劣的*(noninferior)。S 中的一点是有效的当且仅当它的象在 Z 中是非支配的。

定义 3.1 对给定点 $\mathbf{z}^0 \in Z$，它是非支配的(nondominated)当且仅当不存在其他点 $\mathbf{z} \in Z$，使得对于最大化情况有：

$$z_k > z_k^0 \quad 对于某些\ k \in \{1,2,\cdots,q\}$$
$$z_l \geqslant z_l^0 \quad 对于所有其他\ l \neq k$$

如果对于 $\mathbf{z}^0$ 存在其他点 $\mathbf{z}$ 满足上式，则 $\mathbf{z}^0$ 点称作判据空间中的*支配点*(dominated point)。

定义 3.2 对于给定点 $\mathbf{x}^0 \in S$，它是有效的(efficient)当且仅当不存在其他点 $\mathbf{x} \in S$，使得对于最大化情况有：

$$f_k(\mathbf{x}) > f_k(\mathbf{x}^0) \quad 对于某些\ k \in \{1,2,\cdots,q\}$$
$$f_l(\mathbf{x}) \geqslant f_l(\mathbf{x}^0) \quad 对于所有其他\ l \neq k$$

如果对于 $\mathbf{x}^0$ 存在其他点 $\mathbf{x}$ 满足上式，则 $\mathbf{x}^0$ 点在判据空间中是*无效的*(inefficient)。

决策空间中的某点是有效的当且仅当它的象是判据空间中的非支配点。

为了说明上面的定义，考虑下述线性规划问题：

$$\min \quad z_1(x_1,x_2) = x_1 - 3x_2 \tag{3.7}$$
$$\min \quad z_2(x_1,x_2) = 3x_1 + x_2 \tag{3.8}$$
$$\text{s.t.} \quad g_1(x_1,x_2) = x_1 + 2x_2 - 2 \leqslant 0 \tag{3.9}$$
$$g_2(x_1,x_2) = 2x_1 + x_2 - 2 \leqslant 0 \tag{3.10}$$
$$x_1, x_2 \geqslant 0 \tag{3.11}$$

决策空间中可行域 S 用图 3.1 表示。可行区域的极值点是 $\mathbf{x}^1=(0,0)$，$\mathbf{x}^2=(1,0)$，$\mathbf{x}^3=(2/3,2/3)$，$\mathbf{x}^4=(0,1)$。判据空间中的可行区域 Z 通过将 S 集用两个目标(3.7)和(3.8)映射得到，用图 3.2 表示。对应的极值点是 $\mathbf{z}^1=(0,0)$，$\mathbf{z}^2=(1,3)$，$\mathbf{z}^3=(-4/3,8/3)$，$\mathbf{z}^4=(-3,1)$。从图中可以看出两个区域都是凸的，Z 上的极值点是 S 上极值点的象。当问题的目标函数有负系数时，Z 中可能有一部分不在$\mathbb{R}^2$的非负象限内。可行区域 Z 中带斜线的边就是非支配解集合。这个集合可以简单地在可行区域 Z 中通过视觉观察

得到。查看 z^1 和 z^4 之间的点可以知道，随着 $f_2(x_1,x_2)$ 从0增加到1，$f_1(x_1,x_2)$ 从0降低到−3，因此 z^1 和 z^4 之间的点都是非支配点。在决策空间中对应的有效点是 x^3 和 x^4 之间的线段。

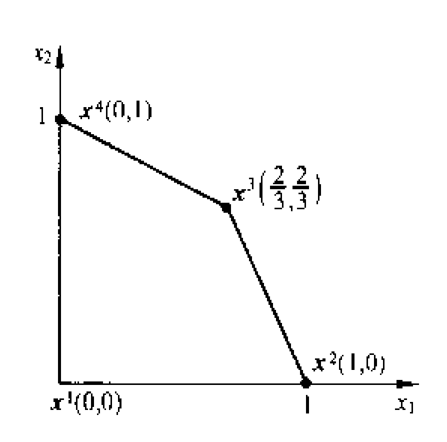


图3.1　决策空间中的可行区域和有效解

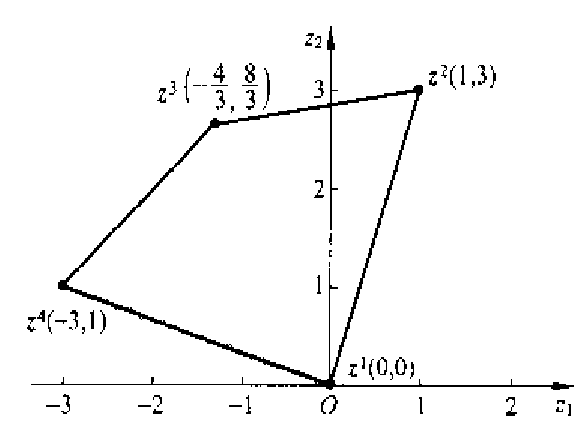


图3.2　判据空间中的可行区域和非支配解

在判据空间中有一个特殊点叫做理想点(ideal point)或正理想解(positive ideal solution)，用 $z^*=(z_1^*,z_2^*,\cdots,z_q^*)$ 来表示，其中 $z_k^*=\sup\{f_k(x)\mid x\in S\}$，$k=1,2,\cdots,q$。点 z^* 称作理想点因为通常该点无法达到。对于每个目标来说，寻找 z_k^* 是可能的。但寻找一个能够同时最大化每个 $f_k(\cdot)$，$k=1,2,\cdots,q$ 的点 z^* 通常是非常复杂的。

负理想解(negative ideal solution)用来表示判据空间中的不利情况，用 $\bar{z}=(\bar{z}_1,\bar{z}_2,\cdots,\bar{z}_q)$ 来表示，其中 $\bar{z}_k=\inf\{f_k(x)\mid x\in S\}$，$k=1,2,\cdots,q$。与正理想点类似，负理想点通常也无法达到。而单个的 $\bar{z}_k$ 是可能达到的。

3.2.2　偏好结构

多目标问题解的基本特征之一是存在一组无法相互进行比较的有效解(efficient solutions)。在实际的决策情况中，通常需要从非支配解中选择一个作为给定问题的最终解。然而，如果不提供对于不同目标附加的偏好信息，很可能无法从解中进行选择。因此，如何从这些可选的非支配解中做出最后的选择本质上依赖于个人主观的偏好。从概念上讲，偏好(preference)是通过采用某人对目标的价值判断来对有效集合中无法进行比较的解给出排序(order)。偏好反映了某人根据对问题事先掌握的知识对所有目标进行的折中或者对某个目标进行的强调。给定了偏好我们就可以将非支配集中可选的解进行排序，然后获得最终解，这就是通常决策过程的结果。这个最终解称作最优妥协解(best-compromised solution)。

偏好通常由二元关系来表示。二元关系(binary relation)就是一组有序对。对于一组给定对 u 和 v，可能且仅可能发生下面的一种关系：

1. u 比 v 好，或对 u 的偏好大于 v，用 $u\succ v$ 来表示。

2. u 比 v 差，或对 u 的偏好小于 v，用 $u \prec v$ 来表示。

3. u 与 v 相当，或 u 与 v 同等偏好，用 $u \sim v$ 来表示。

4. u 和 v 之间的偏好关系未定义，用 $u ? v$ 来表示。

符号 $\succ$，$\prec$，$\sim$ 和 ? 是定义比较和关系的算子。决策空间 Z 中所有点对的偏好关系都可以通过上述 4 种算子中的一个来指定。任何明确的偏好信息可以通过笛卡儿积(Cartesian product) $Z \times Z$ 的子集来表示如下：

$\{\succ\}$——偏好关系集。

$\{\prec\}$——反偏好关系集。

$\{\sim\}$——同等偏好关系集。

$\{?\}$——未定义偏好关系集。

比如，$(u,v) \in \{\succ\}$ 意味着 $u \succ v$。$\{\succ\}$ 集与 $\{\prec\}$ 集是对称(symmetric)的，即 $(u,v) \in \{\succ\}$ 当且仅当 $(v,u) \in \{\prec\}$。因此如果 $\{\succ\}$ 知道了，$\{\prec\}$ 也就知道了，反之亦然。因此偏好结构可以惟一地被 $\{\succ\}$、$\{\sim\}$ 和 $\{?\}$ 确定。

3.2.3 基本求解方法

一般希望决策过程或者获得妥协解或偏好解，或者确定所有非支配解。因此，主要存在两种多目标优化问题的求解方法：(1)*产生式方法*(generating approaches)，(2)*基于偏好的方法*(preference-based approaches)。产生式方法用于确定整个 Pareto 解集或解集的近似。基于偏好的方法试图获得妥协解或偏好解。如果没有对目标偏好结构(preference structure)的先验知识，就只能采用产生式方法来检验所有的非支配可选解。如果有对于目标相对重要性的某些想法，就可以对偏好进行限定。有了偏好信息就可以确定妥协解或偏好解。

产生式方法和基于偏好的方法都有其优缺点。基于偏好的方法需要决策者能够用正式和有结构的方式来清晰表述其偏好，而产生式方法需要决策者从整个 Pareto 解中做选择来进行必要的价值判断。然而产生式方法还存在一些大多数基于偏好的方法没有的其他问题。带有两个或三个目标的问题可以从图的含义中清楚地表明选择。但在高维问题中显示结果和进行选择就非常复杂了，复杂程度随着目标数量的增加而大致成指数上升趋势。计算费用也随着目标数量的增加而显著上升。

从*求解方法*(solution approaches)的角度出发，大多数传统方法将多个目标减少为一个，然后用数学规划工具求解问题。为了用数学规划工具来求解多目标问题，首先需要用数字的形式来表明偏好。数字越大，偏好越强。这种需求促成了各种标量化方法的发展。采用这些方法(诸如效用函数法、权重和法和妥协法等)将多目标优化问题转换为单目标或一系列单目标优化问题，然后可以求解变化后的问题。

权重和方法(weighted-sum approach)　为每个目标函数分配权重并将其组合成为

一个目标函数的基本思想由 Zadeh 首先提出[690]。权重和方法可以表示如下：

$$\max \quad z(\boldsymbol{x}) = \sum_{k=1}^{q} w_k f_k(\boldsymbol{x}) \tag{3.12}$$

$$\text{s.t.} \quad \boldsymbol{x} \in S \tag{3.13}$$

权重可以理解为目标与目标之间的相对重要性或价值。换句话说，权重可以看作我们对目标的偏好。因此问题(3.12)～(3.13)的最优解涉及了一个特定的偏好结构。此外，该问题的最优解在所有权重都是正数的前提下是非支配解。

对于决策空间 S 中两个给定的点 $\boldsymbol{x}^1$ 和 $\boldsymbol{x}^2$，$z(\boldsymbol{x}^1)>z(\boldsymbol{x}^2)$ 当且仅当 $\boldsymbol{x}^1 \succ \boldsymbol{x}^2$，$z(\boldsymbol{x}^1)=z(\boldsymbol{x}^2)$ 当且仅当 $\boldsymbol{x}^1 \sim \boldsymbol{x}^2$，因此有：

$$\{\succ\} = \{(\boldsymbol{x}^1, \boldsymbol{x}^2) \in S \times S \mid z(\boldsymbol{x}^1) > z(\boldsymbol{x}^2)\}$$
$$\{\sim\} = \{(\boldsymbol{x}^1, \boldsymbol{x}^2) \in S \times S \mid z(\boldsymbol{x}^1) = z(\boldsymbol{x}^2)\}$$
$$\{?\} = \varnothing$$

由于权重和方法采用的数值次序，不存在偏好比较中的不明确情况。对于任意两点来说，要么一点好于另一点，要么一点差于另一点，要么二者相当。三种情况必须满足一种。偏好结构中不存在未定义集。

效用函数方法(utility function approach) 效用函数是偏好结构的一种数学表示，它将判据空间中的点映射为实数，数越大则表明其对应的判据空间中点的偏好程度越高。设实值函数 $U(\cdot): Z \to \mathrm{R}$ 为 Z 上带有偏好的效用函数。对于判据空间 Z 中任意两点 $\boldsymbol{z}^1$ 和 $\boldsymbol{z}^2$，$U(\boldsymbol{z}^1)>U(\boldsymbol{z}^2)$ 当且仅当 $\boldsymbol{z}^1 \succ \boldsymbol{z}^2$，$U(\boldsymbol{z}^1)=U(\boldsymbol{z}^2)$ 当且仅当 $\boldsymbol{z}^1 \sim \boldsymbol{z}^2$。于是有：

$$\{\succ\} = \{(\boldsymbol{z}^1, \boldsymbol{z}^2) \in Z \times Z \mid U(\boldsymbol{z}^1) > U(\boldsymbol{z}^2)\}$$
$$\{\sim\} = \{(\boldsymbol{z}^1, \boldsymbol{z}^2) \in Z \times Z \mid U(\boldsymbol{z}^1) = U(\boldsymbol{z}^2)\}$$
$$\{?\} = \varnothing$$

与权重和方法相同的是，由于效用函数给出的是数值次序，不存在偏好比较中的不明确情况。考虑判据空间中任意两点，在偏好、反偏好和同等偏好三种情况中必然有一种代表这两点的关系。

对于给定的偏好结构，如果可以构造效用函数 $U(\cdot)$，则求解多目标优化问题的理想方法就是求解效用函数规划：

$$\max\{U(z_1, \cdots, z_q) \mid z_1 = f_1(\boldsymbol{x}), \cdots, z_q = f_q(\boldsymbol{x}), \boldsymbol{x} \in S\} \tag{3.14}$$

实际上，我们可能永远无法获得偏好结构的特定效用函数的数学表示。鉴于获得效用函数的困难性以及效用函数规划几乎一定的非线性性，这种方法在解决多目标优化问题上有点受限制。然而效用方法从概念上讲是很有用的。关于效用理论的文献是丰富的。Keeney 和 Raiffa 的书中对这种方法及其在现实世界中的各种应用做出了精彩的描述[352]。

妥协方法(compromise approach) 妥协方法可以看作一种根据距离函数进行目标

搜索行为的数学表示。由于该方法简单易懂而且容易计算，它得到了普遍的应用。妥协方法寻找与理想点(ideal point)最近的解。根据先前给出的理想点 z^* 的定义，对于判据向量的每一个元素，我们无法得到比理想点更好的结果。鉴于 z^* 通常无法达到，如果没有其他可选点来解决目标的冲突，就必须进行妥协。给定 $z \in Z$，最终选取 z 而不是找到理想点的后悔函数可以用下面的距离函数来近似：

$$r(z) = \| z - z^* \| \tag{3.15}$$

其中 $\| z - z^* \|$ 是在某种特定范数意义上从 z 到 z^* 的距离。由于 L_p 范数比较清晰，因此经常采用。对于一个给定的正数 $p \geq 1$，有：

$$r(z;\ p) = \| z - z^* \|_p = \left[\sum_{j=1}^{q} | z_j - z_j^* |^p \right]^{1/p} \tag{3.16}$$ [1]

L_p 范数意义下的妥协解(compromise solution)就是最小化 $r(z;p)$ 的点 $z \in Z$，或是最小化 $r(z(x);p)$ 的点 $x \in S$。

后悔函数(regret function) $r(z;p)$ 对于每一个 $|z_j - z_j^*|$ 值的重视程度相同。如果判据具有不同的重要性，可以指派一个权重向量 $w = (w_1, w_2, \cdots, w_q)$ 来表明不同的重要程度。在这种情况下，有下面的加权 L_p 范数：

$$r(z;\ p, w) = \| z - z^* \|_{p,w} = \left[\sum_{j=1}^{q} w_j^p | z_j - z_j^* |^p \right]^{1/p} \tag{3.17}$$

对于参数 p 来说，如果 $p=1$，则所有目标的后悔函数之和得到最大的强调：

$$r(z;\ 1, w) = \sum_{j=1}^{q} w_j | z_j - z_j^* | \tag{3.18}$$

如果 $p=\infty$，则单个目标的后悔函数得到最大的强调：

$$r(z;\ \infty,\ w) = \max\{ w_j | z_j - z_j^* |, j = 1, 2, \cdots, q \} \tag{3.19}$$

一般来说，对 p 的选择反映了对目标中最大后悔函数的关心。p 值越大，则关心程度越大。

带有参数 p 的后悔函数 $r(\cdot)$：$Z \to \mathbb{R}$ 是关于 Z 上妥协偏好的实值函数。对于判据空间 Z 上任意两点 z^1 和 z^2，$r(z^1;p) < r(z^2;p)$ 当且仅当 $z^1 \succ z^2$，$r(z^1;p) = r(z^2;p)$ 当且仅当 $z^1 \sim z^2$。于是有：

$$\{\succ\} = \{(z^1, z^2) \in Z \times Z \mid r(z^1;p) < r(z^2;p)\}$$
$$\{\sim\} = \{(z^1, z^2) \in Z \times Z \mid r(z^1;p) = r(z^2;p)\}$$
$$\{?\} = \varnothing$$

与权重和方法和效用函数方法相同的是，由于后悔函数给出的是数值次序，不存在偏好比较中的不明确情况。考虑判据空间中任意两点，在偏好、反偏好和同等偏好三种情况中必

[1] 译者注：此式表明，妥协方法将对多个目标的最大化问题转化为对后悔函数的最小化问题。

然有一种代表这两点的关系。

字典顺序方法(lexicographic ordering approach) 在这种类型的偏好中，次序比目标更重要。对 $z=(z_1,z_2,\cdots,z_q)$ 进行索引使得对所有 $k=1,2,\cdots,q-1$ 都有第 k 个元素压倒性地比第 $(k+1)$ 个元素重要。字典顺序偏好定义如下：对点 z^1 的偏好大于对点 z^2 的偏好当且仅当下面两种情况有一种发生：$z_1^1>z_1^2$ 或者存在某些 $r\in\{1,2,\cdots,q\}$ 使得 $z_r^1>z_r^2$，同时满足对于 $i=1,2,\cdots,r-1$ 都有 $z_i^1=z_i^2$。不相同的两点的字典顺序一定不同[①]。于是我们有：

$$\{\succ\}=\{(z^1,z^2)\in Z\times Z\mid z^1\ \text{在字典顺序上优于}\ z^2\}$$

$$\{\sim\}=\{(z,z)\in Z\times Z\mid z\in Z\}$$

$$\{?\}=\varnothing$$

Pareto 方法(Pareto approach) Pareto 方法假设关于对目标偏好的任何信息都不存在，我们知道的所有信息就是对于每个元素 z_j，其值越大越得到偏好。Pareto 偏好定义如下：对于判据空间 Z 上任意两点 z^1 和 z^2，点 z^1 偏好于点 z^2 当且仅当 $z^1\geqslant z^2$，即至少存在一个元素 r 满足 $z_r^1>z_r^2$，其他元素满足 $z_k^1\geqslant z_k^2$，$k=1,2,\cdots,q$，$k\neq r$。于是有：

$$\{\succ\}=\{(z^1,z^2)\in Z\times Z\mid z^1\geqslant z^2\}$$

$$\{\sim\}=\{(z,z)\in Z\times Z\mid z\in Z\}$$

$$\{?\}=\{(z^1,z^2)\in Z\times Z\mid \text{既不能满足}\ z^1\geqslant z^2,\text{也不能满足}\ z^1\leqslant z^2\}$$

与其他方法比较，一旦得到恰当的后悔函数、效用函数或权重系数，集合{?}就是空集，问题成为一维比较或数学规划问题。采用 Pareto 方法，我们将处理集合{?}不是空集的问题。

3.2.4 问题的结构和特性

一般来说，多目标优化问题比较复杂，因此求解较困难。有关问题求解的两类复杂性需要明确区分：(1)问题中固有的复杂性，(2)与求解方法有关的复杂性。现有许多求解方法的最大缺点在于它们对权重值、目标给定的次序或效用函数的形状非常敏感。从本质上讲，这种困难是由求解方法引起的，而不是问题本身的困难。

通常多目标优化问题可以通过下列结构和特征来描述：(1)目标函数和判据空间，(2)约束函数和解空间，(3)问题的规模。目标函数和约束函数可能是线性或非线性、凸或凹或不凸不凹、可微或不可微、连续或半连续、单峰或多峰的。因此，判据空间和解空间可能是凸或非凸、紧或非紧、连通或分离的。问题的规模由决策变量、约束和目标的数量

① 译者注：由于对第一个目标的偏好远大于对其他目标的偏好，因此只要 $z_1^1>z_1^2$ 就可以认定 z^1 优于 z^2。如果两点在前 $r-1$ 个目标上均同等重要，而在第 r 个目标上有 $z_r^1>z_r^2$，则也可以认定 z^1 优于 z^2。

决定。传统方法主要被限制在带有线性函数和凸解空间的简单情况上。这种多目标优化问题一般被转换为单目标或一系列单目标优化问题，然后采用扩展的基于梯度或单纯形的方法来求解转换后的问题。

3.3 遗传多目标优化

在过去的20年中，遗传算法作为多目标优化问题的新求解方法受到了相当程度的关注，这就诞生了进化或遗传多目标优化。这个主题已经由 Fonseca 和 Fleming[202]、Horn[303]、Tamaki、Kita 和 Kobayashi[612]进行了综述。本节将简要描述遗传算法不同实现方法中的一些基本思想。某些有代表性的方法将在后续几节中进行详细介绍。

3.3.1 遗传搜索的特征

遗传算法内在的特征说明了为何遗传搜索适合用于多目标优化问题。遗传算法的基本特征是通过在代与代之间维持由潜在解组成的种群来实现多向性和全局搜索。这种从种群到种群的方法在搜索 Pareto 解时是有用的。

遗传算法不需要许多数学上的必备条件，可以处理所有类型的目标函数和约束。由于算法的进化本质，遗传算法可以在不考虑问题特定内部工作方式的前提下用于搜索解。因此能够用遗传算法求解的复杂问题的数量可能比能够用传统算法求解的复杂问题的数量大得多。

由于遗传算法作为一种超启发式算法(metaheuristics)，可以灵活地将传统方法结合进其主框架，因此可以利用遗传算法和传统方法两方面的优势来建立对问题更有效的求解方法。对遗传算法在多目标优化问题中应用不断增长的研究兴趣对数学界提出了巨大的理论和实际上的挑战。

3.3.2 适应值分配机制

遗传算法本质上是一种求解的超策略。当采用遗传算法来求解给定的问题时，有必要对其主要组成部分(比如编码方式、重组算子、适应值分配、选择、约束处理等等)进行改进使其获得对给定问题更有效的实现。由于多目标优化问题是约束和组合优化问题的自然扩展，过去20年里在用遗传算法求解约束优化问题和组合优化问题中开发的许多实用方法和技巧都可以方便地应用于多目标优化问题中。因此，当考虑如何将遗传算法应用于多目标优化问题时，我们仅需考虑与问题相关的一些特殊情况。

用遗传算法求解多目标优化问题中出现的一个特殊情况就是如何根据多个目标来确定个体的适应值。适应值分配机制在过去的10年中得到了广泛的研究，已经提出并测试了若干种方法。这些方法可以粗略地分类如下：

1. 向量评价方法
2. 权重和方法
3. 基于 Pareto 的方法
4. 妥协方法
5. 目标规划方法

根据偏好信息与适应值函数结合程度的不同，这些方法包括从给定全部偏好信息（当直接组合目标函数或将其进行排序时）到不给任何偏好信息（当采用基于 Pareto 的方法时）。另外的一个显著特点是可以先给出一个粗略的偏好，随着进化搜索的进程来对偏好进行持续的细化。偏好的持续细化类似于多目标优化中经常使用的交互式过程，在这种过程中决策者在每次迭代过程中修改偏好。细化机制的独特之处在于偏好是随着进化搜索的过程由某种适应性改善机制来逐渐改善的，而不是由决策者在每次迭代过程中进行干预。当然交互式过程也可以结合进遗传搜索来指导偏好细化。

可能第一个将简单遗传算法扩展为求解多目标优化问题方法的重要工作是*向量评价方法*（vector evaluation approach）[563]。该方法采用了向量形式的适应值度量来产生下一代，而不是使用标量适应值度量方式来评价染色体。对于由 q 个目标的给定问题，每代中的选择过程是一个循环：它重复 q 次，每次循环依次使用一个目标。每次循环用这个目标选出下一代中一部分个体。

有两种*基于 Pareto 的方法*（Pareto-based approach）：*Pareto 排序*（Pareto ranking）和 *Pareto 竞争*（Pareto tournament）。Pareto 基于排序的适应值分配方法由 Goldberg 首先提出[249]。该方法希望对所有 Pareto 个体分配相同的复制概率。它包括两个主要步骤：

1. 基于 Pareto 排序对种群进行分类。
2. 根据排序对个体分配选择概率。

排序过程为：对非支配个体标记为顺序 1，然后从竞争中移去这些个体；在余下的个体中寻找非支配个体并将其标记为顺序 2，等等。在这种方法中，Pareto 排序方法对所有非支配解分配相同的适应值，使其获得相同的复制概率。值得注意的是，大多数其他适应值分配方法对非支配解分配不同的适应值，这与非支配性本身的定义相矛盾。

Pareto 竞争方法由 Horn，Nafpliotis 和 Goldberg 提出[306]。在竞争方法中采用了小生境 Pareto 概念，而不是非支配分类和排序选择。小生境 Pareto 指的是具有最小邻居数量的 Pareto 解赢得竞争。

权重和方法采用与传统多目标优化方法相同的基本前提。该方法为每个目标函数分配权重并将权重目标组合为单一目标函数。从概念上讲，该方法容易理解而且便于计算。只需要合适的权重向量就可以实现该方法。然而该方法存在困难之处。一旦被嵌入遗传算法，这种方法的弱点可以被基于种群和进化搜索的力量弥补。最近提出了若干种权重

调整方法来充分利用遗传搜索的力量：(1)固定权重方法，(2)随机权重方法，(3)适应性权重方法。

在固定权重方法(fixed-weight approach)中，权重在整个进化过程中不会改变。权重可由事先获得的有关目标的知识确定。如果没有方法事先获得这种知识，甚至可以随机确定权重。这种方法由权重对于固定方向给出选择压力。

Murata，Ishibuchi 和 Tanaka 提出了随机权重方法(random-weight approach)[476]。在选择过程中的每一步都随机给出权重，这样可以给所有可能的组合以平等的机会。统计上，这种方法对于 Pareto 边界给出了均匀的选择压力。这种方法忽略了每代的 Pareto 解中可能获得的信息。

Gen 和 Cheng 提出了适应性权重方法(adaptive weight approach)[107,693]。在这种方法中，权重根据当前代进行适应性调整以获得朝向正理想点的搜索压力。由于将当前种群中的有用信息用于每代中重新调整权重，这种方法的选择压力介于固定权重方法和随机权重方法之间。

基于妥协的适应值分配方法(compromise-based fitness assignment method)由 Cheng 和 Gen 在解决双目标最小路径问题时提出[109]。该方法的基本思想和技巧从传统多目标优化中获得。妥协方法通过某种距离的度量来确定与理想解最近的解。正如我们所知的那样，理想解通常无法达到。然而可将它作为可以达到的非支配解的评价标准。如果理想解可以达到，则所有人都会偏好它，因此寻找与理想解最接近的解就成为合理的替代想法。一种常用的邻近程度的度量就是加权 L_p 度量。对于许多复杂问题来说，寻找理想点也是很困难的。为了克服这个困难，就提出了代理理想点(proxy ideal point)的概念来替代理想点。代理理想点是与当前遗传代相关的点，而不是给定问题的点。换句话说，是根据部分已经探索的解空间而不是整个解空间来计算代理理想点。在每代中很容易获得代理理想点。随着进化过程的进行，代理理想点会逐渐接近真实理想点。

目标规划(goal programming)是解决多目标优化问题的强大方法。Gen 和 Liu 将遗传算法应用于求解非线性目标规划问题[237]。该方法采用了基于排序的适应值分配方法来评判个体的价值。由于在目标规划中经常采用字典顺序，因此按照字典方式对个体的目标值进行排序。具体的过程是：根据第一优先目标进行种群排序，如果某些个体具有相同的目标值，根据第二优先目标进行排序，如此进行下去。如果各个目标上两个个体均相同，则随机对其进行排序。然后采用从最好到最差的指数插值进行个体的适应值分配。

从方法论的角度出发，存在两种多目标优化的基本方法：产生式方法和基于偏好法方法。产生式方法用于确定整个 Pareto 解集或解集的近似，而基于偏好方法则试图获得妥协解或偏好解。从概念上讲，向量评价方法、基于 Pareto 排序方法和随机权重方法都属于产生式方法，而目标规划方法则属于基于偏好方法。

在多目标优化中，产生式方法和基于偏好方法都有其优缺点。产生式方法需要决策者从整个Pareto解中进行判断。对于多于3个目标的问题，进行选择非常复杂。这种复杂程度随着目标数量的增加而指数上升。更多的目标也使计算费用显著增加。在遗传多目标优化中，这种情况没有发生根本变化。与之相对的基于偏好方法看来对多目标优化决策者的负担不像产生式方法那样大。原因在于偏好可以在进化过程中逐渐细化，只需要粗略的偏好就可以使进化搜索进行下去。

在多目标优化中，求解方法可以按照产生式方法（寻找整个Pareto解）或者基于偏好方法（寻找偏好解或妥协解）来设计。传统方法不能在一个求解过程中将两种不同的思路实现为同一种求解方法，但在遗传多目标优化中可以这样做。

如何处理不可行解是遗传优化中非常重要的问题，原因在于遗传算子通常会生成不可行解甚至非法解。这个问题在遗传多目标优化中讨论得不那么多，原因在于讨论的焦点都放到如何产生Pareto解上了。大多数测试问题都是小规模人工生成的实例，不带有复杂的约束。对于许多实际应用来说，这就成为相对严重的问题了。原因是在复杂的约束中，不可行解会占整个种群中相对较大的成分[218]。遗传优化实践中应用最广泛的两种方法就是修补方法和惩罚方法。从本质上讲，惩罚方法可以于任何对适应值进行比例变换的方法（如妥协和权重和方法）一起使用。如果需要，可以将修补方法与基于Pareto排序的方法一起使用。

3.3.3　适应值共享和种群多样性

适应值共享（fitness sharing）是维持种群多样性的一种技术。首先需要区分下面两类共享技术：（1）*多峰函数中的共享*（sharing in multimodal functions），（2）*沿着Pareto边界的共享*（sharing along the Pareto frontier）。适应值共享由Goldberg和Richardson针对多峰函数优化提出[248]。后来共享方法在遗传多目标优化中进行扩展，目的是沿着非支配解边界维持种群。

遗传算法用于求解多峰函数的一个问题是，由于选择过程中随机误差的作用，有限的种群最终会收敛到一个最优解上。这种现象称作*遗传漂移*（genetic drift）。共享技术的目的就是防止遗传漂移并促进均匀采样。换句话说，就是在搜索空间的不同峰中分布种群，每个峰根据其高度的比例得到种群中一部分个体。为了达到这个目的，就产生了适应值降低（fitness degradation）的概念。如果某个峰有太多的个体，该峰中所有个体的适应值都将降低以减小其复制能力。

共享函数（sharing function）是根据某个体周围的拥挤程度来确定其个体适应值降低程度的方式。两个个体之间距离的度量可以在两个不同的空间中定义，于是产生两种类型的适应值共享：*基因型共享*（genotypic sharing）和*表现型共享*（phenotypic sharing）。基因型共享中两个个体之间的距离在编码空间度量；表现型共享中距离在解码空间度

量。Deb 的工作指出，一般来说，表现型共享比基因型共享要好[151]。

设 s_i 表示一个用编码形式表示的字符串或染色体。可以直接获得字符串上的度量：

$$d_{ij} = d(s_i, s_j) \tag{3.20}$$

使用字符串上度量的适应值共享称作基因型共享。设 x_i 表示一个用解码形式表示的解或个体。可以得到在决策变量空间(即解空间)上的度量：

$$d_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{3.21}$$

使用解空间上度量的适应值共享称作表现型共享。在此之后定义共享函数 Sh(・)为度量值 d_{ij} 的函数。下面的幂函数(power law function)通常被用作共享函数：

$$\mathrm{Sh}(d_{ij}) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma_{\text{share}}}\right)^{\alpha}, & \text{如果 } d_{ij} < \sigma_{\text{share}} \\ 0, & \text{否则} \end{cases} \tag{3.22}$$

其中 α 是常数。σ_{share}是小生境半径(niche radius)，由用户根据所期望的个体之间最小分离程度事先估计出来。在共享半径距离之内的个体将相互影响适应值的降低程度。

一旦距离度量和共享函数确定了，一个个体的共享后适应值(shared fitness)f_i'就由个体适应值 f_i 除以其小生境数(niche count)m_i 来确定：

$$f_i' = \frac{f_i}{m_i} \tag{3.23}$$

给定个体 i 的小生境数 m_i 由在整个种群中对共享函数进行求和得到：

$$m_i = \sum_{j=1}^{pop_size} \mathrm{Sh}(d_{ij}) \tag{3.24}$$

求和包括个体自身。因此，如果一个个体仅独自存在于一个小生境中($m_i=1$)，它共享后的适应值就是其原始适应值。否则，共享函数根据相邻点的数量和接近程度来降低适应值。

适应值共享最初是与适应值比例选择方式联合使用的。当共享与竞争选择联合使用时，遗传算法表现出混沌性质。这种小生境子种群中的混乱振荡可由 Oei, Goldberg 和 Chang 提出的连续更新共享(continuously updated sharing)来加以避免[488]。这种方法中小生境数不仅由当前种群计算出来，还受下一代部分填充种群的影响。

在遗传多目标优化中，共享方法可以分为下面两类：(1)目标空间中的共享(sharing on the objective space)，(2)解空间中共享(sharing on the solution space)。目标空间中的共享试图得到整个 Pareto 集合上的均匀采样；解空间中的共享试图在容许的解集中进行均匀采样。目标空间中的采样由 Fonseca 和 Fleming[201]提出。在多峰函数优化中，共享在解空间中进行。通常需要关于有限峰数量的先验知识和解空间中小生境均匀分布的假设。对于收敛来说，许多个体根据局部最优解适应值的比例被保留在局部最优解上。与此相反，在多目标优化中情况则不同。不再存在任何局部峰。Pareto 边界上所有点都

是同等优秀的。我们更为关心维持一组全局非支配解，最好它们能够在全局折中表面(global trade-off surface)上均匀地相互隔开并且有代表性。采用排序方法会迫使搜索惟一地集中于全局最优解上。通过在目标值域的非支配个体上，而不是决策变量域上实现适应值共享，可以期望能够实现在全局折中表面上非支配个体的均匀分布。

解空间中的共享由 Srinivas 和 Deb 提出[581]。其主要关注点是沿着 Pareto 边界(Pareto frontier)的多样性可能不对应着解空间中的多样性，而解空间中的多样性对于决策者来说很重要。通过根据下式计算两个个体之间的共享函数来实现共享：

$$\mathrm{Sh}(d_{ij}) = \begin{cases} 1-\left(\dfrac{d_{ij}}{\sigma_{\text{share}}}\right)^2, & \text{如果 } d_{ij} < \sigma_{\text{share}} \\ 0, & \text{否则} \end{cases} \tag{3.25}$$

其中 d_{ij} 是两个个体之间的表现型距离，而 σ_{share} 是两个个体成为同一小生境成员所允许的最大表现型距离。

如果希望种群在多个最优区域中分布，交配限制(mating restriction)就成为另一个值得考虑的问题。交配限制假设相邻的个体在几何上是相似的，因此它们的交配可以形成稳定的小生境。在多峰优化中很容易理解：不同局部峰所在的区域一般具有非常不同的遗传表示。因此交配限制可能帮助杂交算子在局部最优解周围实现微调。多目标优化中，在 Pareto 集合相对小的区域中两个个体具有相似基因型表示的特点不明显。Fonseca 和 Fleming 实现了基于目标域中个体间距离的交配限制。然而，在多目标优化中采用交配限制并不普遍[202]。

3.3.4 Pareto 解的概念

在严格意义上，遗传算法中使用的术语 Pareto 解(Pareto solution)与传统方式的含义不同。在最初的定义中，一个点被称作 Pareto 解当且仅当它是关于给定问题判据空间 Z 中所有点的非支配点。在某些遗传算法中(并非所有)，每代都需要确定 Pareto 解。由于每代种群中仅包含原始问题的部分解，因此 Pareto 解仅表示关于所有当前获得的解的非支配点的含义。某一代中的非支配解可能被以后中产生的新解支配。因此对于遗传算法给定的某一代，从中获得的 Pareto 解可能是问题真正的 Pareto 解，但也可能不是。无法保证遗传算法能够产生给定问题的 Pareto 解。但遗传算法会提供对 Pareto 解很好的近似。

在进化过程中如何维持一组非支配解是多目标优化的特殊问题。原则上有两种不同的处理 Pareto 解的方式：(1)从种群池(population pool)中独立地保持 Pareto 解，(2)不带有保持机制。这两种方式将导致遗传算法实现上两种不同的结构。

现有的大多数方法在每代中确定 Pareto 解，并且将其仅用于计算每个染色体的适应值或对染色体进行排序。没有机制能够确保进化过程中产生的 Pareto 解一定进入下一

代。换句话说，一些 Pareto 解可能在进化过程中丧失了。为了避免这种采样误差，许多研究人员提出了针对 Pareto 解的保持机制[214,319,476]。一个为了保持 Pareto 解而设计的特别池子被加入到遗传算法的基本结构中。在每一代中，Pareto 解集通过删除所有被支配解和加入新产生的 Pareto 解而更新。这种方法的整体结构如下所示：

Pareto 遗传算法过程

```
begin
    t←0;
    初始化 P(t);
    计算 P(t)的目标函数值;
    得到 Pareto 解集合 E(t);
    计算 P(t)的适应值;
    while 不满足停止条件 do
    begin
        对 P(t)进行杂交;
        对 P(t)进行变异;
        计算 P(t)的目标函数值;
        更新 Pareto 解集合 E(t);
        计算 P(t)的适应值;
        从 P(t)中选择 P(t+1);
        t←t+1;
    end
end
```

如果没有保持机制，Pareto 解只能在最后一代中获得。如果优化方法具有物种形成(speciation)的趋势(正如 Schaffer 指出的那样[563])，在很长遗传代数以后整个种群会向着个体最优区域收敛。保持机制在某种程度上通过在每代中保持 Pareto 解来削弱物种的形成。

3.4 向量评价遗传算法

Schaffer 在了解到遗传算法在求解多目标优化潜力的基础上，于 1985 年将简单遗传算法扩展为能够包容不可比较的多个目标的算法，即向量评价遗传算法(vector-evaluated genetic algorithms VEGA)[563]。在 VEGA 方法中，每代中选择过程成为一个循环。在每次循环的过程中根据每个目标选出若干下一代中的优秀个体(即子种群)。然后整个种群被完全打乱并执行杂交和变异算子。这样做的目标是为了促进不同子种群个体之间的交配。通过监视进化过程中的种群来确定非支配个体，但这项信息在 VEGA 中并没有应用。这种方法保护了单个目标上最优秀个体的生存，同时为那些在多于一个目标上好于

平均适应值的个体提供了合理的被选择概率。该方法的过程可以用图3.3说明。

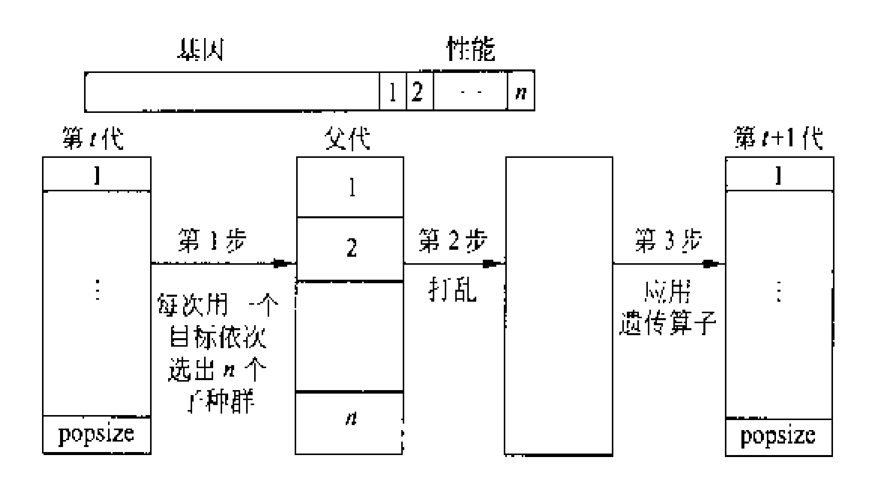


图3.3 VEGA选择

Schaffer通过修改Grefenstette的GENESIS程序[265]实现了VEGA。该算法中，选择过程中的循环针对每个目标选出下一代中的一部分个体。每个父代期望产生子代的数量是每个父代在每个目标上期望产生子代的数量之和。由于Schaffer采用了适应值比例分配选择方式，这也对应于目标本身的比例选择。给定个体最终的整体适应值取决于各目标的线性组合函数，该函数的权重依赖于种群在每代中的分布。这样做的结果就是不同的非支配解个体一般被分配不同的适应值，这一点与其他权重和方法类似。值得注意的是没有个体有明确的整体适应值表示，原因是VEGA不使用标量适应值函数。我们采用这种说法只是用来说明这种算法与常规权重和方法的选择结果很相似，都是由一个个体根据其对应的适应值复制出许多后代。

Schaffer采用了一个简单的两目标单变量问题来测试VEGA的性质：

$$\min \quad f_1 = x^2$$
$$\min \quad f_2 = (x-2)^2$$
$$\text{s.t.} \quad x \in \mathbb{R}^1$$

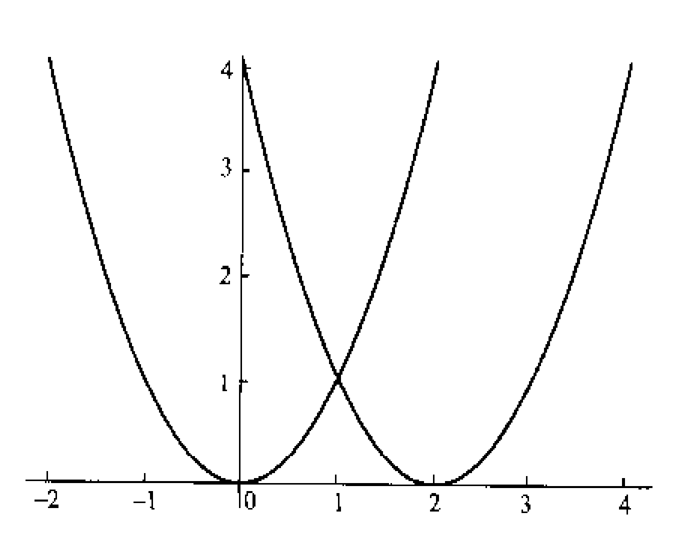


图3.4 与x相对应的目标

图3.4将变量限制在[−2,4]区间内绘出了两个目标的形状。很明显Pareto解由所有在[0,2]区间中的变量x组成。图3.5绘出了问题的判据空间。

在Srinivas和Deb给出的关于VEGA的模拟结果中[581]，遗传算法的参数如下：最大代数500，种群规模100，字符串长度(二进制编码)32，杂

交概率 1.0，变异概率 0。变量 x 的初始区域为[－10，10]。初始种群用图 3.6 显示。第 10 代种群向着非支配区域收敛，如图 3.7 所示。第 100 代几乎整个种群都收敛到非支配解的区域，如图 3.8 所示。注意到第 500 代，种群仅收敛到 3 个子区域，如图 3.9 所示。这就是物种形成（speciation）现象。

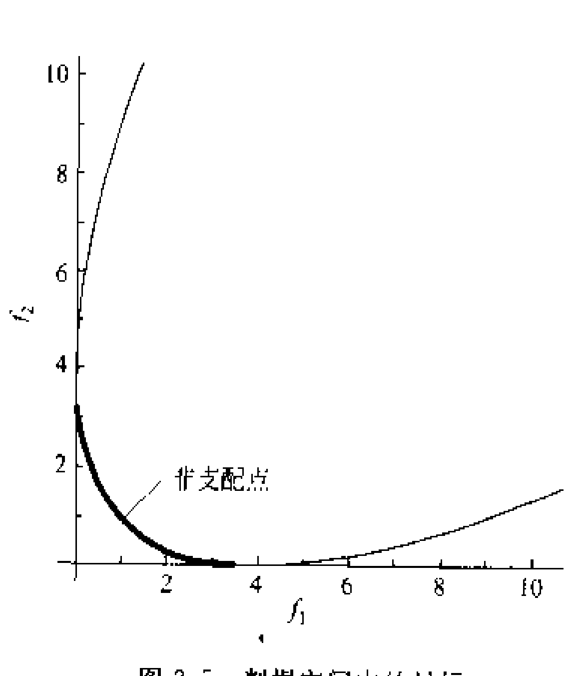


图 3.5 判据空间中的目标

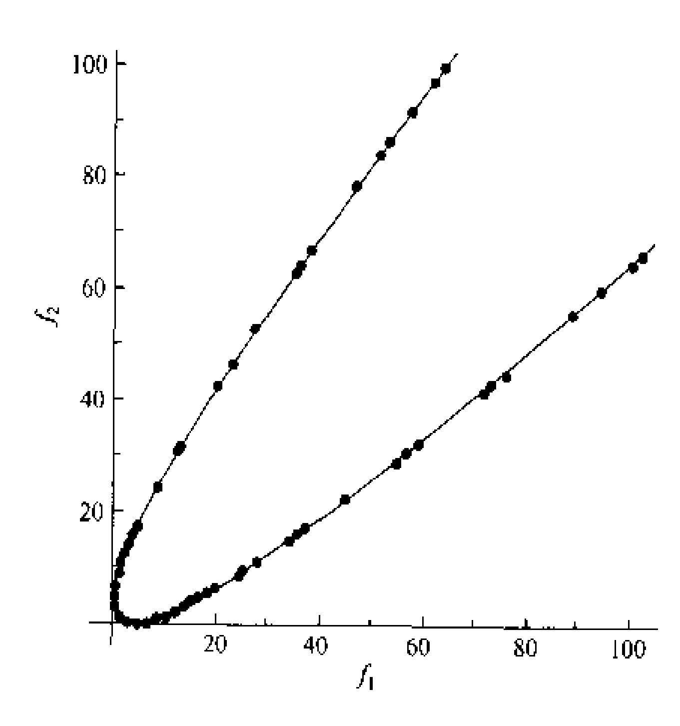


图 3.6 采用 VEGA 在第 0 代获得的种群

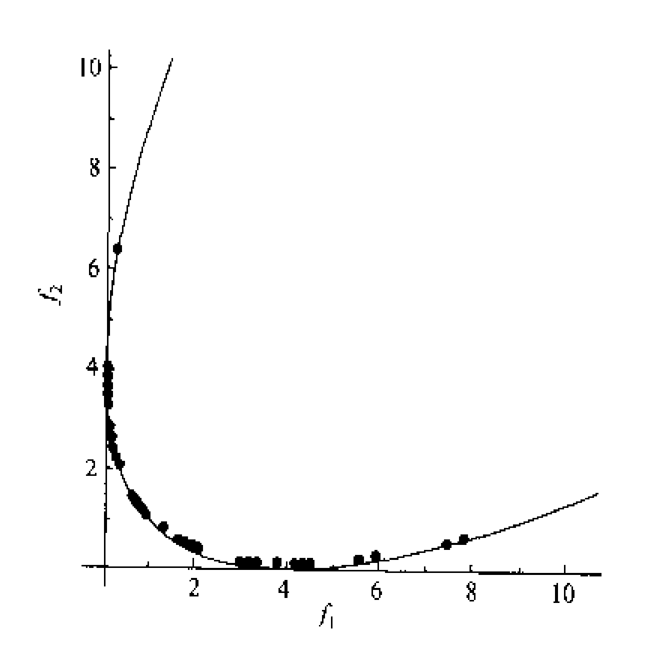


图 3.7 采用 VEGA 在第 10 代获得的种群

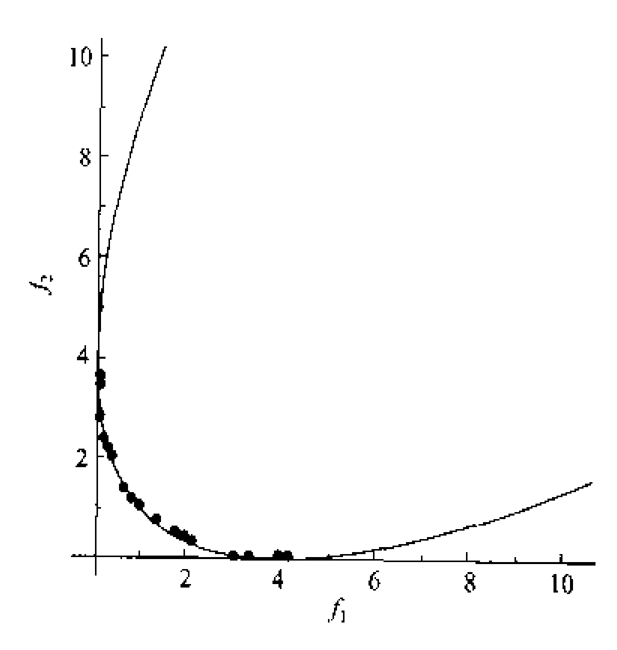


图 3.8 采用 VEGA 在第 100 代获得的种群

Schaffer的工作通常被看作用遗传算法求解多目标优化的开创性努力。虽然VEGA无法给出多目标优化问题满意的解，它还是为开发其他新遗传算法的实现提供了许多有益的提示。大多数以后的工作将VEGA作为比较其性能的范例。

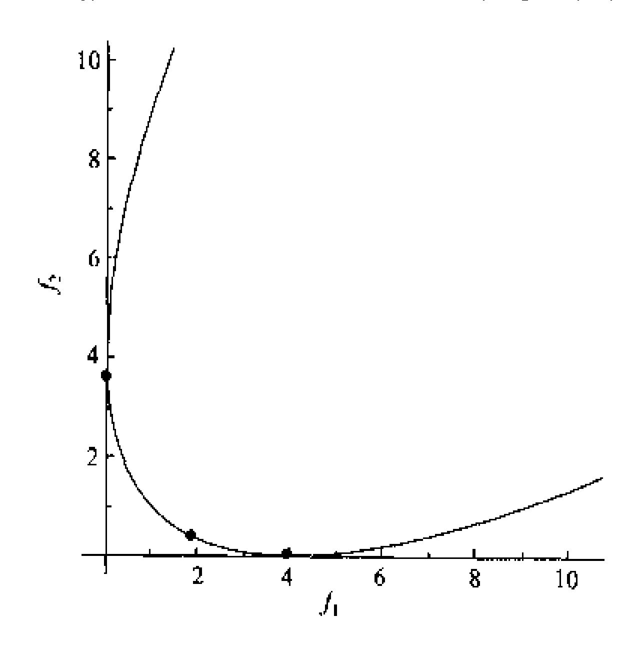


图3.9 采用VEGA在第500代获得的种群

Fourman也采用非集合方法(non-aggregating approach)处理多目标优化问题[205]。他给出了算法两种实现方式。在第一种方式中，事先给目标分配不同的优先权，选择过程根据目标的优先权来进行个体的比较，较高优先权的目标先进行比较。如果两个个体不分胜负，则采用第二高优先权的目标来比较，如此进行下去。正如先前提到的那样，这就是字典顺序(lexicographic ordering)。在第二种方式中，每次比较中随机选取一个目标。据报道，这种方法的表现令人惊讶地好。采用随机选择目标方式产生的最终种群比采用确定性选取目标方式产生的种群表现出更大的可变性。Kursawa实现了一种几乎与Fourman方法相同的目标随机选择算法[383]。

3.5 Pareto排序和竞争方法

3.5.1 Pareto排序方法

Pareto基于排序的适应值分配方法首先由Goldberg提出[249]，目的是使所有Pareto个体得到相同的复制概率。该过程类似于Baker的排序选择(ranking selection)过程[36]，但种群是根据非支配个体进行排序。排序的过程如下：对当前种群中的非支配个体分配次序1，并将其从竞争中移去；然后从当前种群中选出非支配个体并对其分配次序2。该过程持续到种群中所有个体都分配到次序后结束。图3.10用一个简单实例(对两个最小化目标同时进行优化)说明了该过程。

Baker的排序选择很直截了当：

1. 从最好到最差对种群进行排序。
2. 根据排序对个体分配选择概率。

如果设 p_k 表示排序后种群中第 k 个个体的选择概率，Baker的线性排序方法采用下

面的形式：

$$p_k = q - (k-1) \times r$$

其中参数 q 是最好个体的选择概率。设 q_0 是最差个体的选择概率，K 是种群中最后一个次序的数值。参数 r 可以用下式确定：

$$r = \frac{q - q_0}{K - 1}$$

中等个体的适应值根据其次序的比例从 q 降低到 q_0。将 q_0 设为 0 提供了最大的选择压力。

Fonseca 和 Fleming 提出了存在少许不同的 Pareto 排序方法（Pareto ranking method）[201]。当前种群中所有非支配个体分配次序 1，任何其他个体所分配的次序数等于支配该个体的数量加 1。这种方法可以用图 3.11 说明。该方法根据个体的次序对种群进行排序。如果次序相同，顺序随机选取。适应值根据种群中最好到最差个体进行线性或非线性插值的结果来分配，具有相同次序个体的适应值一样。选择方式则采用了 Baker 提出的随机通用采样（stochastic universal sampling）方法[36]。

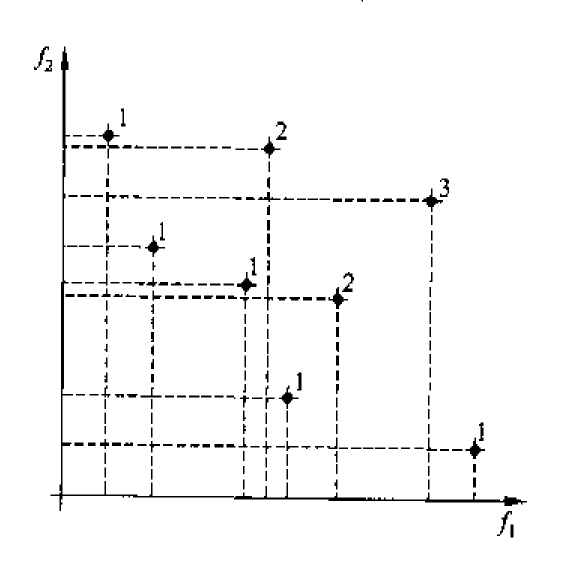


图 3.10 Goldberg 提出的排序方法

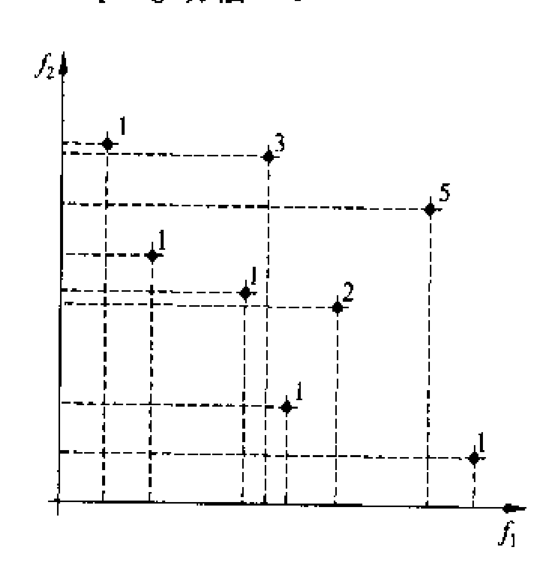


图 3.11 Fonseca 和 Fleming 提出的排序方法

Srinivas 和 Deb 实现了类似的分类和适应值分配方法[581]。种群根据 Goldberg 的 Pareto 排序方法来进行排序。非支配解首先被确定，然后被分配一个很大的哑适应值。为了维持种群多样性，这些个体用它们的哑适应值进行共享。在共享之后，这些非支配个体暂时被忽视。从余下的种群中确定第二批非支配个体，然后它们被分配一个比先前非支配个体共享后最小适应值小一点的哑适应值。该过程一直持续到整个种群都被分为若干类。采用随机剩余比例选择（stochastic remainder proportionate selection）来复制出新一代[76]。该算法的流程如图 3.12 所示。

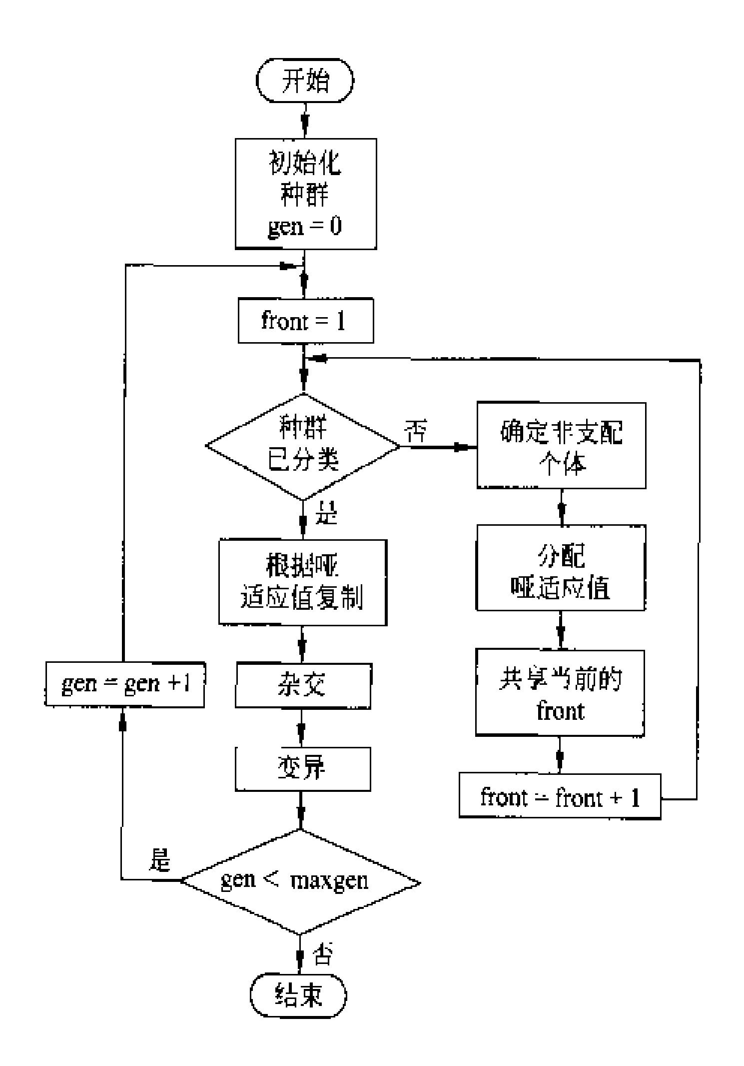


图 3.12 Srinivas 和 Deb 提出的非支配分类方法的流程图

3.5.2 Pareto 竞争方法

Pareto 竞争方法由 Horn，Nafpliotis 和 Goldberg 提出[306]。这种方法从种群中随机选出两个候选个体。同时还从种群中随机选出一组比较个体集。然后每个候选个体与比较集中的每个个体进行比较。可能产生两种结果：

1. 如果一个候选个体被比较集支配，而另一个不被支配，则非支配候选个体被选出来进行复制。

2. 如果两个候选个体都被比较集支配或都支配比较集，则采用共享方法来选择优胜者。

正如我们所知，共享方法根据个体的小生境数来确定个体适应值降低程度。Horn 等人提出了一种新的共享方法。该方法不根据小生境数进行适应值任何形式的降低，而是将具有较小小生境数的个体选为优胜者。小生境数通过求候选个体周围一定距离内种

群中个体的数量来确定。

图3.13说明了这种共享方法在两个非支配个体间的工作过程。这是一个最大化第1个目标并最小化第2个目标的实例。两个候选个体均在Pareto解集中，因此不被比较集支配。从 Pareto 解的观点出发，两个候选个体之间没有偏好。如果希望维持种群多样性，很显然最好选择具有较小小生境数的个体。在该实例中就是候选个体 2。

由于非支配性是将候选个体和从种群中随机选出的比较集进行比较来确定的，该算法成功与否就取决于比较集规模这个参数。如果这个参数太小，该过程仅选出种群中少许非支配个体。如果这个参数太大，则可能发生早熟收敛。

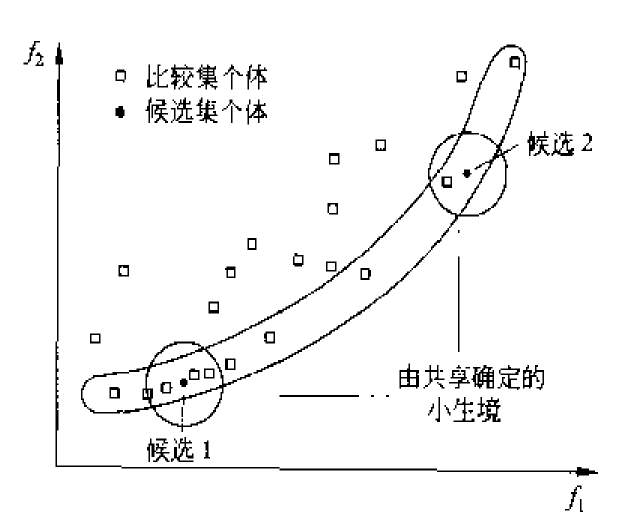


图 3.13 Horn 提出的共享方法

3.6 权重和方法

从概念上讲，权重和方法可以看作是将多目标优化中采用的方法向遗传算法进行扩展。该方法给每个目标函数分配权重，然后将加权目标组合为单一目标函数。事实上，在遗传算法中使用的权重和方法与在传统多目标优化算法中使用的权重和方法在本质上有很大不同。在多目标优化中，权重和方法用来获得妥协解。使算法运行的惟一要求是合适的权重向量。对于给定问题，通常很难获得一组合适的权重。在遗传算法中，最初权重和方法用来使遗传搜索向着 Pareto 前沿面进行。随着进化的进行，权重适应性地重新调整。因此并不一定需要良好的权重向量来使遗传算法运行。另外，权重和方法在传统多目标优化中的缺点也可以被遗传算法基于种群搜索和进化搜索的力量削弱。最近提出了3 种权重设置的方法：固定权重方法、随机权重方法和适应性权重方法[312]。固定权重方法可以看作是对传统标量化方法的模仿，而随机权重方法和适应性权重方法用来更全面地利用遗传算法的搜索能力。正是由于遗传算法内在的基于种群的进化搜索能力才使得这两种方法能够运行。

3.6.1 随机权重方法

Murata，Ishibuchi 和 Tanaka 提出了一种随机权重方法(random-weight approach)来获得目标是 Pareto 前沿面的可变搜索方向[319,476]。存在两种目标空间中有代表性的搜索行为：固定方向搜索和多方向搜索，如图 3.14 和图 3.15 所示。固定权重方法使遗传算

法有向着判据空间中一个固定点所在的区域进行采样的趋势，而随机权重方法则使遗传算法具有可变搜索方向，即在整个 Pareto 前沿面上进行均匀采样的能力。

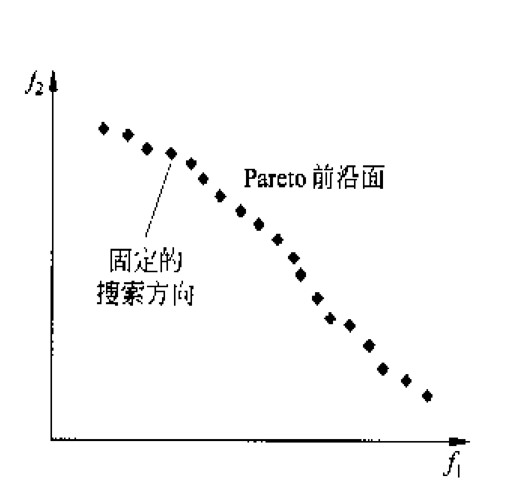


图 3.14　在判据空间中沿着固定方向搜索

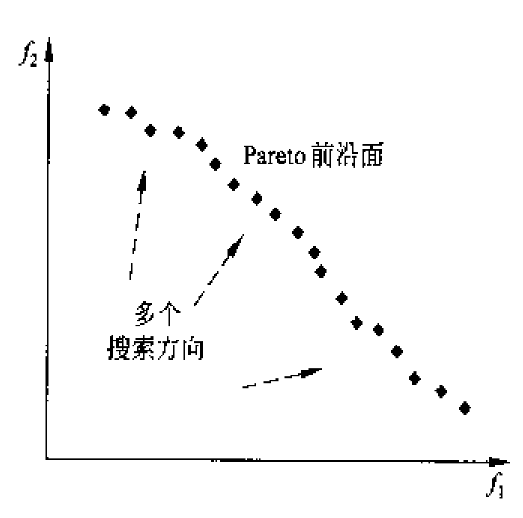


图 3.15　在判据空间中沿着多个方向搜索

假设我们要最大化 q 个目标函数。权重和目标(weighted-sum objective)如下所示：

$$z = \sum_{k=1}^{q} w_k f_k(\boldsymbol{x}) \tag{3.26}$$

随机权重 w_k 由下式计算：

$$w_k = \frac{r_k}{\sum_{j=1}^{q} r_j}, \quad k = 1,2,\cdots,q \tag{3.27}$$

其中 r_i 是非负随机数。

在选出进行杂交的个体对之前，由式(3.27)生成一组新的随机权重，根据式(3.26)计算每个个体的适应值。第 i 个个体的被选择概率 p_i 由下面的线性比例变换函数定义：

$$p_i = \frac{z_i - z_{\min}}{\sum_{j=1}^{pop_size} (z_j - z_{\min})} \tag{3.28}$$

其中 $z_{\min}$ 是当前种群中最差个体的适应值。

每一代临时存储一组 Pareto 解并按代更新。对于带有 q 个目标的问题，Pareto 解中存在 q 个极限点，每一个最大化一个目标。作者提出了精华保留策略(elite preserving strategy)，将 n 个解(包括极限点与一些随机选择的 Pareto 解)直接放入下一代种群。设 N_{pop} 表示种群规模，N_{elite} 表示保存的精华解数量。作者提出的算法的整体结构如下所示：

第 1 步：初始化(initialization)　随机产生包括 N_{pop} 个个体的初始种群。

第 2 步：评价(evaluation)　对于每个个体计算 q 个目标函数值。更新临时 Pareto 解集。

第 3 步：选择(selection)　重复下面的步骤来选出$(N_{\text{pop}} - N_{\text{elite}})$对父代：用式(3.27)指定

随机权重，用式(3.26)计算适应值，用式(3.28)计算被选择概率，为杂交操作选出一对父代个体。

第4步：杂交(crossover) 对于选出的每一对个体，执行杂交操作产生后代。

第5步：变异(mutation) 对杂交操作产生的每个个体执行变异操作。

第6步：最优性策略(elitist strategy) 从临时 Pareto 解集中随机选择 N_{elite}个个体。将选出的 N_{elite}个个体添加到前面步骤产生的($N_{pop}-N_{elite}$)个个体中以构成 N_{pop}个个体的种群。

第7步：终止测试(termination test) 如果事先指定的停止条件得到满足，终止算法。否则返回第2步。

3.6.2 适应性权重方法

Gen 和 Cheng 提出了适应性权重方法(adaptive weight approach)，该方法利用当前种群中一些有用的信息来重新调整权重，从而获得朝向正理想点的搜索压力[107,214~216,677,693]。不失一般性，考虑下面带有 q 个目标的最大化问题：

$$\max \quad \{z_1 = f_1(\boldsymbol{x}),\ z_2 = f_2(\boldsymbol{x}), \cdots, z_q = f_q(\boldsymbol{x})\} \tag{3.29}$$

$$\text{s.t.} \quad g_i(\boldsymbol{x}) \leqslant 0, \quad i = 1,2,\cdots,m \tag{3.30}$$

对于每代中待检查的解来说，在判据空间中定义两个极限点：最大极限点 $\boldsymbol{z}^+$ 和最小极限点 $\boldsymbol{z}^-$ 如下：

$$\boldsymbol{z}^+ = \{z_1^{\max}, z_2^{\max}, \cdots, z_q^{\max}\} \tag{3.31}$$

$$\boldsymbol{z}^- = \{z_1^{\min}, z_2^{\min}, \cdots, z_q^{\min}\} \tag{3.32}$$

其中 $z_k^{\max}$ 和 $z_k^{\min}$ 是当前种群中第 k 个目标的最大值和最小值。设 P 表示当前种群集合。每个目标的最大值和最小值定义如下：

$$z_k^{\max} = \max\{f_k(\boldsymbol{x}) \mid \boldsymbol{x} \in P\}, \quad k = 1,2,\cdots,q \tag{3.33}$$

$$z_k^{\min} = \min\{f_k(\boldsymbol{x}) \mid \boldsymbol{x} \in P\}, \quad k = 1,2,\cdots,q \tag{3.34}$$

由两个极限点定义的超平行四边形(hyperparallelogram)是包含当前所有解的最小超平行四边形。两个极限点每代更新。最大极限点最终将接近正理想点。目标 k 的适应性权重用下式计算：

$$w_k = \frac{1}{z_k^{\max} - z_k^{\min}}, \quad k = 1,2,\cdots,q \tag{3.35}$$

对于给定个体 $\boldsymbol{x}$，权重和目标函数由下面的公式确定：

$$z(\boldsymbol{x}) = \sum_{k=1}^{q} w_k (z_k - z_k^{\min}) \tag{3.36}$$

$$= \sum_{k=1}^{q} \frac{z_k - z_k^{\min}}{z_k^{\max} - z_k^{\min}} \tag{3.37}$$

$$= \sum_{k=1}^{q} \frac{f_k(\boldsymbol{x}) - z_k^{\min}}{z_k^{\max} - z_k^{\min}} \tag{3.38}$$[1]

由于极限点每代更新，权重也随之更新。式(3.39)～式(3.42)定义了由当前解中极限点确定的超平面：

$$(z_1^{\max},\ z_2^{\min},\cdots,\ z_k^{\min},\cdots,\ z_q^{\min}) \tag{3.39}$$

$$(z_1^{\min},\ z_2^{\max},\cdots,\ z_k^{\min},\cdots,\ z_q^{\min}) \tag{3.40}$$

$$\cdots$$

$$(z_1^{\min},\ z_2^{\min},\cdots,\ z_k^{\max},\cdots,\ z_q^{\min}) \tag{3.41}$$

$$(z_1^{\min},\ z_2^{\min},\cdots,\ z_k^{\min},\cdots,\ z_q^{\max}) \tag{3.42}$$

超平面将判据空间 Z 划分为两个半空间：一个半空间包括正理想点，用 Z^+ 表示，另一个半空间包括负理想点，用 Z^- 表示。所有的 Pareto 解存在于空间 Z^+ 中，所有 Z^+ 空间中点的适应值比 Z^- 空间中点的适应值要大。由于最大极限点随着进化过程接近正理想点，超平面将最终接近正理想点。因此适应性权重方法可以根据当前种群来重新调整权重，从而获得朝向正理想点的搜索压力。

对于最小化的情况，我们可以仅将原始问题转换为等价的最大化问题，然后应用式(3.38)。对于最大化问题，式(3.38)可以简化如下：

$$z(\boldsymbol{x}) = \sum_{k=1}^{q} w_k z_k \tag{3.43}$$

$$= \sum_{k=1}^{q} \frac{z_k}{z_k^{\max} - z_k^{\min}} \tag{3.44}$$

$$= \sum_{k=1}^{q} \frac{f_k(\boldsymbol{x})}{z_k^{\max} - z_k^{\min}} \tag{3.45}$$

让我们看一个双目标最大化问题的例子：

$$\max \quad \{z_1 = f_1(\boldsymbol{x}),\ z_2 = f_2(\boldsymbol{x})\} \tag{3.46}$$

$$\text{s.t.} \quad g_i(\boldsymbol{x}) \leqslant 0, \quad i = 1,2,\cdots,m \tag{3.47}$$

对于给定的一代，两个极限点确定如下：

$$z_1^{\max} = \max\{z_1(\boldsymbol{x}^j),\ j = 1,2,\cdots,pop_size\} \tag{3.48}$$

$$z_2^{\max} = \max\{z_2(\boldsymbol{x}^j),\ j = 1,2,\cdots,pop_size\} \tag{3.49}$$

$$z_1^{\min} = \min\{z_1(\boldsymbol{x}^j),\ j = 1,2,\cdots,pop_size\} \tag{3.50}$$

$$z_2^{\min} = \min\{z_2(\boldsymbol{x}^j),\ j = 1,2,\cdots,pop_size\} \tag{3.51}$$

适应性权重(adaptive weight)计算如下：

[1] 译者注：式(3.33)权重定义和式(3.36)括号中减去 $kz_k^{\min}$ 的目的是：使个体 x 对应的加权目标函数值归一化到[0,q]区间上。

$$w_1 = \frac{1}{z_1^{max} - z_1^{min}} \tag{3.52}$$

$$w_2 = \frac{1}{z_2^{max} - z_2^{min}} \tag{3.53}$$

于是权重和目标函数由下式给出：

$$z(\boldsymbol{x}) = w_1 z_1 + w_2 z_2 \tag{3.54}$$

$$= w_1 f_1(\boldsymbol{x}) + w_2 f_2(\boldsymbol{x}) \tag{3.55}$$

由极限点(z_1^{max}, z_2^{min})和(z_1^{min}, z_2^{max})定义的适应性移动线如图 3.16 所示。由极限点(z_1^{max}, z_2^{max})和(z_1^{min}, z_2^{min})定义的矩形是包含当前所有解的最小矩形。

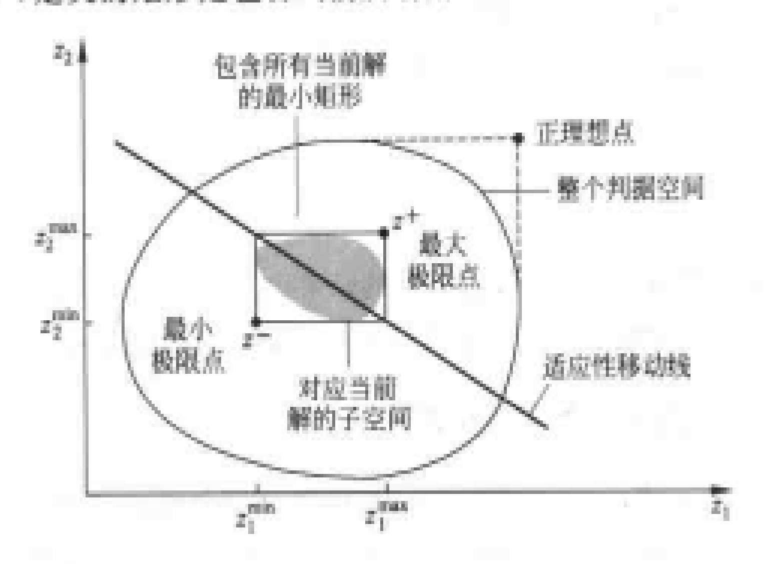


图 3.16 适应性权重和适应性超平面

由于用来操作染色体的遗传算子常常产生不可行后代，因此遗传多目标优化中的一个重要问题就是如何处理约束。Gen 和 Cheng 提出了适应性罚方法来处理不可行个体[216,218]。在当前种群 $P(t)$ 中给定一个个体 $\boldsymbol{x}$，其*适应性罚函数*(adaptive penalty function)构造如下：

$$p(\boldsymbol{x}) = 1 - \frac{1}{m}\sum_{i=1}^{m}\left(\frac{\Delta b_i(\boldsymbol{x})}{\Delta b_i^{max}}\right)^{\alpha} \tag{3.56}$$

其中：

$$\Delta b_i(\boldsymbol{x}) = \max\{0, g_i(\boldsymbol{x}) - b_i\} \tag{3.57}$$ ①

$$\Delta b_i^{max} = \max\{\varepsilon, \Delta b_i(\boldsymbol{x}) \mid \boldsymbol{x} \in P(t)\} \tag{3.58}$$

其中 $\Delta b_i(\boldsymbol{x})$是当前染色体对第 i 个约束的违背值，Δb_i^{max} 是当前种群中对约束 i 的最大违背值，ε 是一个小正数，用来避免罚函数中出现被零除的情况。对于高度约束的最优化问题，每代中不可行解占据相对较大的部分。该罚函数在每代中适应性调整惩罚率，从而既

① 译者注：对应着约束 $g_i(x) \leqslant b_i$，$i = 1,2,\cdots,m$ 的情况。

保存了不可行解有用的信息，又对不可行解施加了选择压力，还避免了过渡惩罚。带有罚函数的适应值函数具有下面的形式：

$$\mathrm{eval}(\boldsymbol{x}) = z(\boldsymbol{x})p(\boldsymbol{x}) \tag{3.59}$$

适应性权重方法的整体过程归纳如下：

第1步：初始化(initialization) 随机产生初始种群。

第2步：评价(evaluation) 对于每个个体计算目标值、惩罚值和适应值。

第3步：Pareto集(Pareto set) 更新Pareto解集。

第4步：选择(selection) 使用轮盘赌方法选出下一代。

第5步：产生(production) 用杂交和变异产生后代。

第6步：终止(termination) 如果达到最大遗传代数，终止算法。否则返回第2步。

3.7 距离方法

Osyczka和Kundu提出了用于计算适应值的*距离方法*(distance method)[495,496]。这种方法的基本思想是根据每个个体到前一代获得的Pareto解之间的距离来分配其适应值。

3.7.1 距离方法的一般思想

该方法中采用了潜在值(potential value)的概念。这个概念是仅分配给每个Pareto解的标量值，与给定Pareto解的适应值不同。更新完Pareto集后，对所有Pareto解分配相同的潜在值，这样就使得每个新产生的个体能够分配合理的适应值。

正如图3.17所示，现存的Pareto解可能有不同的潜在值，用(p_1，p_2，…，p_5)表示。

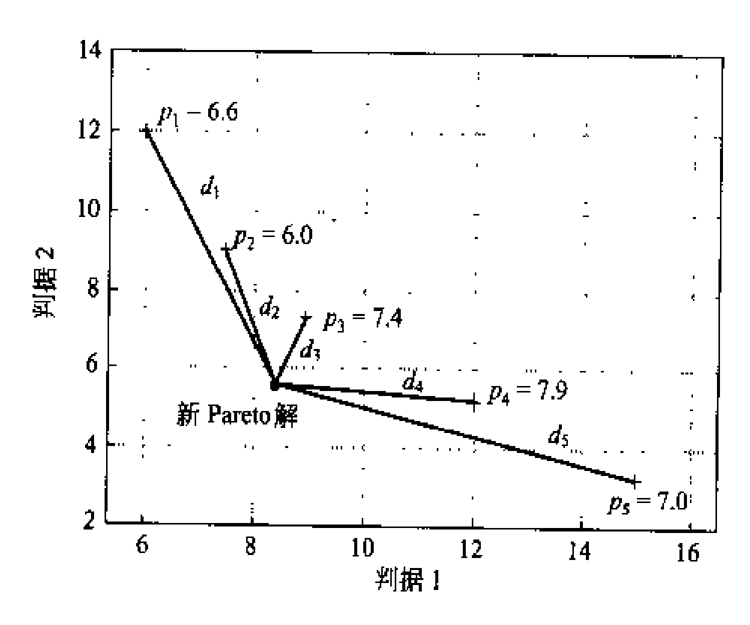


图3.17 第1种情况判据空间中潜在值和距离的度量

对于新产生的解，到所有现存 Pareto 解之间的距离用($d_1,d_2,\cdots,d_5$)来表示。其中 d_3 具有最小距离，则该距离用来计算新解的适应值。

新产生的个体可能属于下列 3 种类型之一：

1. 是一个 Pareto 解，支配一些或所有当前的 Pareto 解。
2. 是一个 Pareto 解，但不支配任何当前的 Pareto 解。
3. 是一个解，至少被一个现存的 Pareto 解支配。

在第 1 种情况中，新解的适应值用最大潜在值和最小距离之和来确定。然后通过将被支配解从 Pareto 解集移去，并将新解添加入该解集。新解的潜在值与其适应值相同。设 f 表示新解的适应值，p 表示现存 Pareto 解的数量，有

$$f = p_{\max} + d_{\min}$$

其中

$$p_{\max} = \max\{p_i \mid i = 1,2,\cdots,p\}$$

$$d_{\min} = \min\{d_i \mid i = 1,2,\cdots,p\}$$

在图 3.17 中，新解支配潜在值为 p_3 的解。新解的适应值是 $f=p_{\max}+d_{\min}=p_4+d_3$。被支配的个体从 Pareto 解集中移去。

第 2 种情况中，新解的适应值用最近 Pareto 解的潜在值和最小距离之和来确定。新解的潜在值与其适应值相同，并将其加入 Pareto 解集中。如图 3.18 所示，新解是一个新的 Pareto 解。它的适应值为 $f=p_4+d_4$，潜在值为$p_6=f$。

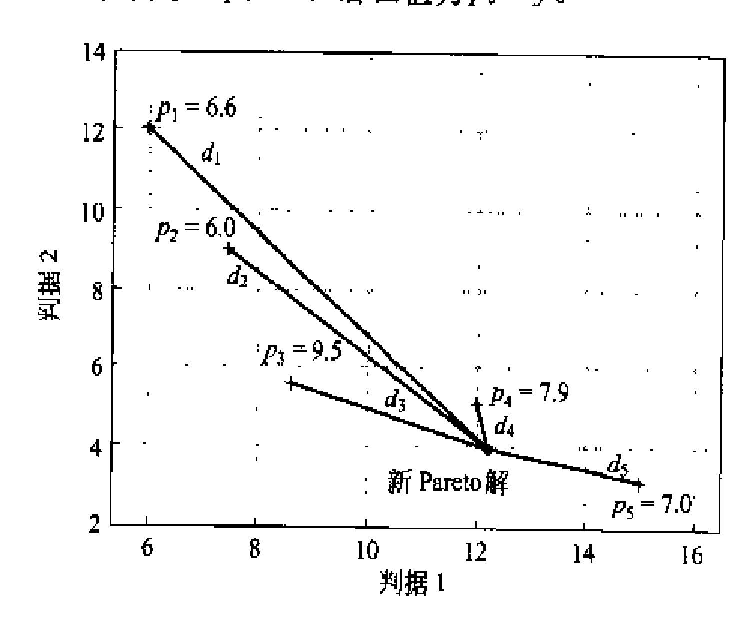


图 3.18 第 2 种情况判据空间中潜在值和距离的度量

第 3 种情况中，新解的适应值由与它最近 Pareto 解的潜在值减去最小距离来确定。如图 3.19 所示，新解的适应值是 $f=p_4-d_4$。

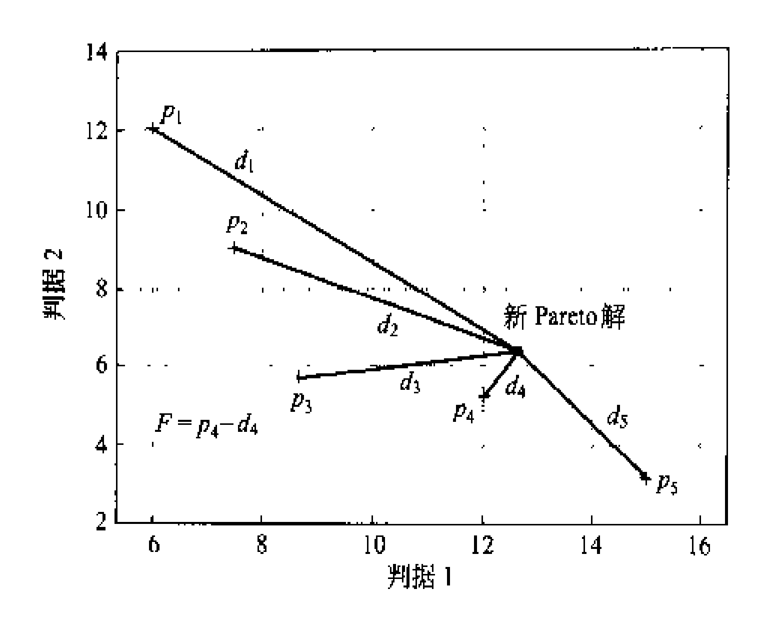


图 3.19　第3种情况判据空间中潜在值和距离的度量

3.7.2 计算距离度量

考虑下面一般形式的非线性多目标优化问题：

$$\min \quad \{f_1(x), f_2(x), \cdots, f_q(x)\} \tag{3.60}$$

$$\text{s.t.} \quad g_i(\boldsymbol{x}) \geqslant 0 \quad i = 1,2,\cdots,m_0 \tag{3.61}$$

$$h_i(\boldsymbol{x}) = 0 \quad i = 1,2,\cdots,m_1 \tag{3.62}$$

Osyczka 和 Kundu 采用如下所示的外部罚函数将该问题转换为无约束问题[495]。

$$\phi_k(\boldsymbol{x}) = f_k(\boldsymbol{x}) + r\sum_{i=1}^{m_1}[h_i(\boldsymbol{x})]^2 + r\sum_{i=1}^{m_0}G_i[g_i(\boldsymbol{x})]^2, \quad k = 1,2,\cdots,q \tag{3.63}$$

其中 G_i 是 Heaviside 算子，满足对于 $g_i(\boldsymbol{x})>0, G_i=0$，以及对 $g_i(\boldsymbol{x})\leqslant 0, G_i=1$。$r$ 是正乘子，用来控制惩罚项的幅度。

假设遗传搜索找到了 p 个 Pareto 解。设 f_{kl} 表示第 l 个 Pareto 解中第 k 个目标函数的值。对于用遗传算子产生的新解 $\boldsymbol{x}$，它到所有现存 Pareto 解之间的距离用下面的公式计算：

$$d_l(\boldsymbol{x}) = \sqrt{\sum_{k=1}^{q}\left(\frac{f_{kl}-\phi_k(\boldsymbol{x})}{f_{kl}}\right)^2}, \quad l = 1,2,\cdots,p \qquad (3.64)^{①}$$

① 译者注：这是表现型上的距离，而不是基因型上的距离。即当前点与 Pareto 解的目标函数值越接近，则其计算出来的距离值越小。

用于计算适应值的最小距离用下面的公式计算：

$$d_{l^*} = \min\{d_l(x) \mid l = 1,2,\cdots,p\} \tag{3.65}$$

其中 l^* 表示与新解 x 最近的现存 Pareto 解。

设 t 代表当前的遗传代数，J 是第 t 代产生的解的数量，j 是第 t 代产生的解的下标，T 是最后一代。距离方法的整体过程总结如下：

第 1 步：设 $t \leftarrow 1, j \leftarrow 1$。第一个产生的解成为 Pareto 解，其潜在值是 p_1。p_1 是任意选择的值，称作起始潜在值。

第 2 步：$j \leftarrow j+1$，检查是否满足 $j \leqslant J$。如果满足，进入第 3 步；否则进入第 5 步。

第 3 步：产生一个新个体 x 并用式(3.64)计算它到所有现存 Pareto 解的相对距离，然后用式(3.65)找出其中的最小距离 $d_{l^*}(x)$。

第 4 步：用下面方式将新解与所有现存的 Pareto 解进行比较：

(4.1) 如果新解是一个 Pareto 解，并且支配至少一个当前的 Pareto 解。用下面的式子计算其适应值：

$$F = p_{\max} + d_{l^*}(x) \tag{3.66}$$

$p_{\max} \leftarrow F$。更新 Pareto 解集。新解的潜在值是 F。返回第 2 步。

(4.2) 如果新解是一个 Pareto 解，但不支配所有当前的 Pareto 解。用下面的式子计算其适应值：

$$F = p_{l^*} + d_{l^*}(x) \quad (3.67)^{①}$$

将其添加到 Pareto 解集中，潜在值为 F。如果 $F > p_{\max}$，$p_{\max} \leftarrow F$。返回第 2 步。

(4.3) 如果该解不是一个新的 Pareto 解，用下面的式子计算其适应值：

$$F = p_{l^*} - d_{l^*}(x) \tag{3.68}$$

如果 $F < 0, F \leftarrow 0$ 从而避免适应值为负。返回第 2 步。

第 5 步：将现存所有 Pareto 解的潜在值替代为 $p_{\max}$，即

$$p_l = p_{\max}, \quad l = 1,2,\cdots,p \tag{3.69}$$

第 6 步：$t \leftarrow t+1$。如果 $t > T$，终止算法；否则检查最近若干代中是否有任何改进(代数需要事先指定)。如果存在改进，$j \leftarrow 0$，返回第 2 步；否则终止算法。

需要说明的是，距离方法对于 p_1 和 r 的设置比较敏感。对于任何不可行解，r 的值越高，$d_l(x)$ 所表示的距离就越高。因此适应值最终接近 0②。如果太多个体的适应值为

① 译者注：p_{l^*} 是最近 Pareto 解的潜在值。

② 译者注：由式(3.63)可知，不可行解的目标函数值随着 r 的增加而增加。由式(3.64)可知，当 r 很大时，x 到第 l 个 Pareto 解之间的距离也很大。由式(3.68)可知，由于 r 很大，不可行解成为被支配解，很可能 $F<0$。根据规则，该解的适应值为 0。

0，遗传搜索将无法进行。另外，如果 p_1 的初始值太小，式(3.68)将趋向0适应值。另一方面，如果初始潜在值太大，不同解之间适应值的差别会很不明显。这将导致选择压力过小或个体的均匀分布，结果导致遗传算法收敛得非常缓慢。

3.7.3 距离方法的应用

Osyczka 和 Kundu 采用下面的问题测试了距离方法[496]。

$$f_1(\boldsymbol{x})=-25(x_1-2)^2-(x_2-2)^2-(x_3-1)^2-(x_4-4)^2-(x_5-1)^2$$

$$f_2(\boldsymbol{x})=(x_1-1)^2+(x_2-1)^2+(x_3-1)^2+(x_4-1)^2+(x_5-1)^2$$

$$\begin{aligned}
\text{s.t.}\quad & g_1(\boldsymbol{x}) = x_1+x_2-2\geqslant 0\\
& g_2(\boldsymbol{x}) = 6-x_1-x_2\geqslant 0\\
& g_3(\boldsymbol{x}) = 2+x_1-x_2\geqslant 0\\
& g_4(\boldsymbol{x}) = 2-x_1+3x_2\geqslant 0\\
& g_5(\boldsymbol{x}) = 4-(x_3-3)^2-x_4\geqslant 0\\
& g_6(\boldsymbol{x}) = (x_5-3)^2+x_6-4\geqslant 0\\
& 10\geqslant x_i\geqslant 0,\quad i=1,2,3,4,5,6
\end{aligned}$$

采用二进制字符串对每个决策变量进行编码。惩罚乘子 r 选为1000，初始潜在值 p_1 选为10。最大遗传代数设为50，种群规模设为400。采用距离方法和随机多目标优化方法获得的Pareto解的比较参见图3.20，从图中可知遗传算法可以得到更好的解。图3.21显示了随着进化过程Pareto边界的移动。

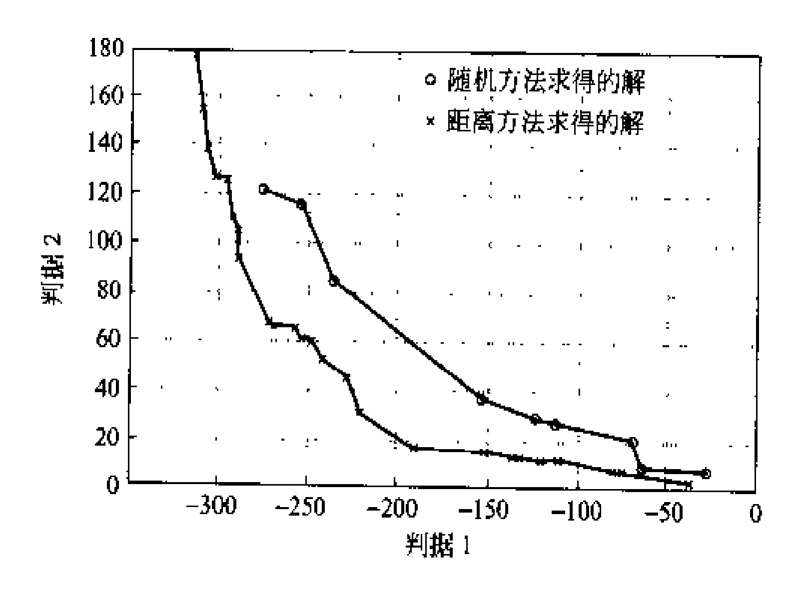


图3.20　对获得的Pareto解的比较

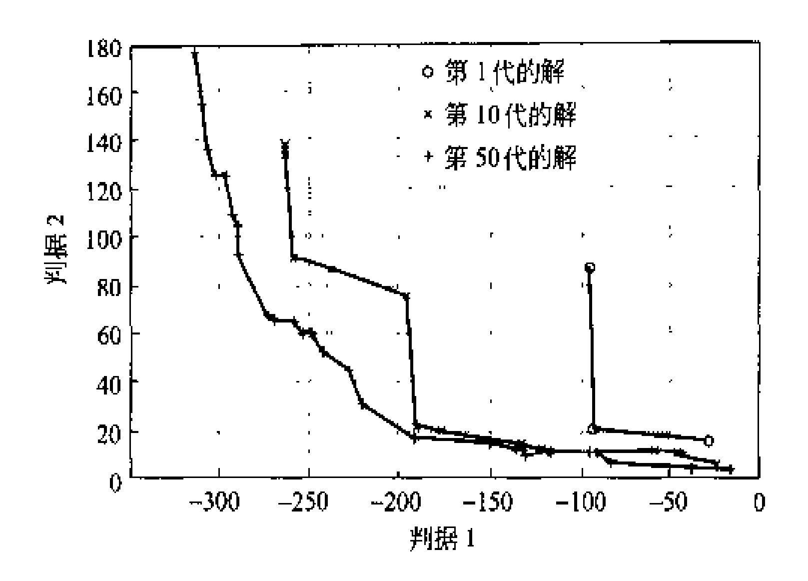


图 3.21 随着进化过程 Pareto 边界的移动

3.8 妥协方法

基于妥协解的适应值分配由 Cheng 和 Gen 提出，该方法可以获得一个妥协解，而不是产生所有 Pareto 解[109]。对于许多实际问题，Pareto 解集可能很大，可能与问题规模成指数关系。因此在最坏情况下求解所需的计算工作就会随着问题规模的增加而指数上升。需要对一大群 Pareto 解进行评价并从中选出最好的一个会成为决策者难以承受的问题。因此在这种情况下，决策者对获得整个 Pareto 解集并不感兴趣。与产生式方法不同，*妥协方法*(compromise approach)寻找妥协解来克服这个困难。

妥协方法根据某种距离度量方式来确定与理想解最近的解。*加权 L_p 范数*(weighted L_p-norm)被用作距离度量方式。

$$r(z;p,w) = \| z - z^* \|_{p,w} = \left[\sum_{j=1}^{q} w_j^p \mid z_j - z_j^* \mid^p \right]^{1/p} \tag{3.70}$$

其中$(z_1^*, z_2^*, \cdots, z_q^*)$是判据空间 Z 中的正理想点，权重$(w_1, w_2, \cdots, w_q)$被分配给目标并用来强调其不同的重要程度。正如我们所知，理想解实际上无法达到。然而它可以很好地成为评价可达到的非支配解的标准。对于许多复杂问题来说，寻找理想点也是困难的任务。为了克服这个困难，就用代理理想点的概念来替代理想点。代理理想点是当前代中相应的理想点，而不是给定问题的理想点。设 P 表示当前种群集，代理理想点$(z_1^{\min}, z_2^{\min}, \cdots, z_q^{\min})$根据下式计算：

$$z_1^{\min} = \min\{z_1(\boldsymbol{x}) \mid \boldsymbol{x} \in P\} \tag{3.71}$$

$$z_2^{\min} = \min\{z_2(\boldsymbol{x}) \mid \boldsymbol{x} \in P\} \tag{3.72}$$

…

$$z_q^{\min} = \min\{z_q(x) \mid x \in P\} \tag{3.73}$$

代理理想点在每代中很容易获得。随着进化过程，代理理想点会最终接近真实理想点。

考虑最小化问题。由于后悔值越小个体越好，因此需要将后悔值转换为适应值从而确保优秀个体具有较大的适应值。设 $r(x)$ 表示个体 x 的后悔值，$r_{\max}$ 表示当前代中的最大后悔值，$r_{\min}$ 表示当前代中的最小后悔值。变换方式如下：

$$\text{eval}(x) = \frac{r_{\max} - r(x) + \gamma}{r_{\max} - r_{\min} + \gamma} \tag{3.74}$$

其中 γ 是正实数，通常被限制在开区间(0,1)中。该系数有两个作用：(1)避免式(3.74)产生被零除错误，(2)可以将选择方式从适应值比例选择调整到纯粹随机选择。①

3.9 目标规划方法

目标规划方法(goal programming approach)的基本思想是给每个目标(objective)函数建立明确的数值目标(goal)，然后寻找一个其目标(objective)函数值到对应目标(goal)的偏差之加权和最小的解[98,222,315,403]。从本质上讲，存在两种目标规划问题。一种是无优先权目标规划(nonpreemptive goal programming)，基本上所有目标都具有相同的重要性。另一种是优先权目标规划(preemptive goal programming)，目标中存在优先级别的层次，因此具有第一重要性的目标得到第一优先关注，具有第二重要性的目标得到第二优先关注，依此类推。当前讲目标规划主要指的是优先权目标规划。这种方法已经成为解决多目标决策问题基本并且被广泛认可的方法。

非线性目标规划(nonlinear goal programming)是一种数学规划方法，用来解决那些具有冲突目标的约束问题。用户针对每个目标函数(objective function)给出需要达到的目标(goal)和目标(goal)达到的优先顺序。该方法按照给定的顺序寻找满足尽可能多目标(goal)的最优解。非线性目标规划的一般表达式如下：

$$\min \quad z_0 = \sum_{k=1}^{q} \sum_{i=1}^{m_0} p_k (w_{ki}^+ d_i^+ + w_{ki}^- d_i^-) \tag{3.75}$$

$$\text{s.t.} \quad f_i(x) + d_i^- - d_i^+ = b_i, \quad i = 1,2,\cdots,m_0 \tag{3.76}$$

$$g_i(x) \leqslant 0, \quad i = m_0 + 1,\cdots,\overline{m}_1 (= m_0 + m_1) \tag{3.77}$$

$$h_i(x) = 0, \quad i = \overline{m}_1 + 1,\cdots,m (= \overline{m}_1 + m_2) \tag{3.78}$$

---

① 译者注：如果 $\gamma=0$，在不考虑被零除错误的前提下，根据式(3.74)计算出的适应值在[0,1]之间分布。最优秀的个体(后悔值为 $r_{\min}$)的适应值为1，最差个体(后悔值为 $r_{\max}$)的适应值为0。这对应着最大选择压力的情况。随着 γ 值的增大，最好个体和最差个体之间适应值的差距逐渐减小，选择压力逐渐降低。另外，如果算法接近收敛(即种群中所有个体的后悔值都相似)，则 $r_{\max} \approx r_{\min} \approx r(x)$，这时基于适应值比例的选择方式表现出随机选择的特性。

$$d_i^-, d_i^+ \geqslant 0, \ i = 1,2,\cdots,m_0 \tag{3.79}$$

其中

p_k——第 k 个优先权值(对于所有 k 都满足 $p_k \gg p_{k+1}$)

d_i^+——代表第 i 个目标(goal)超过预期的正偏差变量

d_i^-——代表第 i 个目标(goal)不足预期的负偏差变量

w_{ki}^+——在优先权 p_k 上分配给 d_i^+ 的正权重

w_{ki}^-——在优先权 p_k 上分配给 d_i^- 的负权重

x——n 维决策向量

f_i——在目标约束上从$\mathbb{R}^n$到$\mathbb{R}^1$的映射

g_i——在不等式约束上从$\mathbb{R}^n$到$\mathbb{R}^1$的映射

h_i——在等式约束上从$\mathbb{R}^n$到$\mathbb{R}^1$的映射

b_i——目标 i 的目标(goal)值或期望级别

q——优先权数量

m_0——目标约束的数量

m_1——实不等式约束的数量

m_2——实等式约束的数量

有时目标函数如下所示：

$$\text{lexmin}\left\{z_1 = \sum_{i=1}^{m_0}(w_{1i}^+ d_i^- + w_{1i}^- d_i^-), \cdots, z_q = \sum_{i=1}^{m_0}(w_{qi}^+ d_i^+ + w_{qi} d_i^-)\right\} \tag{3.80}$$

其中 lexmin 是按照字典顺序最小化目标的意思。

Gen 和 Liu 提出了用于解决非线性目标规划问题的遗传算法[236]。该算法采用了基于排序的适应值分配方法来评价每个染色体的价值。由于目标规划的目标函数间惯用字典顺序，个体按照目标函数的字典顺序来进行排列。然后采用从最好个体到最差个体的指数插值函数来对每个个体的适应值进行分配。评价过程包括 3 步：

第 1 步：根据式(3.80)计算目标值(objective value)。每个个体存在 q 个目标函数值，即

$$\left\{\sum_{i=1}^{m_0}(w_{1i}^+ d_i^- + w_{1i}^- d_i), \sum_{i=1}^{m_0}(w_{2i}^+ d_i^- + w_{2i}^- d_i^-), \cdots, \sum_{i=1}^{m_0}(w_{qi}^+ d_i^+ + w_{qi}^- d_i)\right\}$$

第 2 步：根据第一优先级目标函数值 $\sum_{i=1}^{m_0}(w_{1i}^+ d_i^+ + w_{1i}^- d_i)$ 对个体进行排序。如果某些个体具有相同的目标函数值，则它们根据第二优先级目标函数值进行排序，如此进行下去。如果某些个体最终无法排序，则让它们随机排序。这样个体就按照目标函数值增加的顺序排列起来。

第 3 步：为每个个体分配基于排序的适应值。设 r_k 是个体 x_k 的顺序。根据用户给定的

一个参数 $a\in(0,1)$定义基于排序的适应值函数如下：

$$\text{eval}(\boldsymbol{x}_k)=a(1-a)^{r_k-1}$$

其中 $r_k=1$ 意味着最好的个体，而 $r_k=pop_size$ 意味着最差的个体。我们有[①]

$$\sum_{k=1}^{pop_size}\text{eval}(\boldsymbol{x}_k)\approx 1$$

Taguchi，Ida 和 Gen 提出了用遗传算法解决带有间隔系数的非线性目标规划问题的方法[607]。对于给定个体 $\boldsymbol{x}$，其适应值采用下面的权重和形式：

$$\text{eval}(\boldsymbol{x})=\sum_{k=1}^{q}\sum_{i=1}^{m_0}p_k(w_{ki}^{+}d_i^{+}+w_{ki}^{-}d_i^{-}) \tag{3.81}$$

其中 p_k 是优先权值，按照下式计算：

$$p_k=10^{2*(q-k)} \tag{3.82}$$

注意到式(3.81)的特点是：适应值越小，个体越好。因此个体根据适应值升序排序。下一代从中选出前 pop_size 个不相同个体。

最近 Gen 和 Liu 开发了用于解决最优容量扩展问题的进化算法[235]，Taguchi，Gen 和 Ida 报道了用遗传算法解决多非线性整数规划问题[733]，Kim 等人报道了用遗传算法解决多目标装配线平衡问题[724]，Deb 和 Goyal 针对机械元件设计问题开发了基于遗传适应性搜索的灵活优化方法[712]。

① 译者注：$\sum[a+a(1-a)+\cdots+a(1-a)^{N-1}+\cdots]=a/[1-(1-a)]=1$。

第4章 模糊优化问题

4.1 引言

大多数优化问题可以描述为数学规划问题。数学规划一般用于不同级别管理和制造等学科中的规划和决策，但这种情况下一般采用清晰的目标函数和约束。实际问题不仅在表达利润和损失的目标函数中允许存在偏差，在表达可能投资规模等约束中也允许存在偏差。对这种情况更为理想的表示需要妥善处理这种不明确性。已经提出了不同形式的模糊数学规划(fuzzy mathematical programming)来处理这种情况，其中包括模糊线性数学规划、模糊非线性数学规划、模糊整数规划、模糊混合整数规划、模糊多目标规划等[349,547,550,618,706]。许多文章(参考 Luhandjula[430,431]和 Yazenin[680])考虑了模糊线性规划或模糊多目标线性规划问题，并且提出了一系列利用 Zadeh 提出的可能性理论[691]将原始约束转换为清晰等价式的思想。已有学者提出不带线性假设的机会约束规划、多目标规划和目标规划的一般理论性框架[418,419]。

遗传算法在各种数学规划问题中有许多成功的应用。不幸的是，遗传算法在清晰数学规划问题中的实现不能直接应用于解决模糊规划问题。必须采取某些技巧，这些技巧能够在进化过程中处理模糊性，从而用遗传算法解决模糊数学规划问题。

本章将介绍一些遗传算法在典型模糊优化问题中的成功应用，比如模糊线性规划问题、模糊非线性规划问题、模糊非线性混合整数目标规划问题以及模糊多目标整数规划。为了便于读者深入理解，还介绍了一些数值例子。

4.2 模糊线性规划

自 Bellman 和 Zadeh 提出模糊决策概念以来，关于模糊数学规划领域的研究成为热点[52,390,408,429,616,655,706,707]。根据 Bellman 和 Zadeh 的定义，模糊决策(fuzzy decision)就是用最大最小算子定义的模糊目标和模糊约束的汇合。基于这种定义，Zimmermann 开发了对于对称模糊线性规划模型的容差方法(tolerance approach)。这种方法是第一个可以解决带有模糊约束和目标的线性规划问题的实用方法。该方法在用于解决诸如制造规划、资源分配等实际问题方面有相当大的潜力。自此，模糊线性规划(fuzzy linear programming)向许多方向发展，实现了许多成功的应用。模糊规划被认为是模糊环境下多目标优化的重要组成部分。7.6 节中带模糊系数的双目标运输问题就是现实世界中典

型的模糊多目标优化问题。

本节首先通过一个简单模糊线性规划的数值例子来对线性规划的基本概念进行综述，然后介绍一种用于解决目标和资源模糊的线性规划问题的非精确方法。这种方法不寻找精确的最优解，而是采用沿着加权梯度方向变异的遗传算法寻找具有可接受隶属度的一组非精确解。然后通过人机接口方式，决策者可以从这组解中通过凸组合的方式找到其偏好的解。最后用数值试验表明可以用这种新方法得到令人满意的结果。

4.2.1　模糊线性规划模型

首先将下面的简单生产计划问题作为可以用线性规划求解问题的例子。

例4.1　某工厂下个月需要生产两种类型的产品A和B。每单位A产品和B产品的净利润分别是2百万美元和3百万美元。生产A和B需要劳动时间和一种关键材料。每单位A产品需要4人月劳动时间和2单位关键材料；每单位B产品需要2人月劳动时间和4单位关键材料。有8人月劳动时间和8单位关键材料可供支配（参见表4.1）。在给定劳动时间和材料限制之后，管理者需要知道为了最大化总利润，需要生产多少单位的A产品和多少单位的B产品。这个生产计划问题可以用公式表示为下面的线性规划问题，该问题最大化线性利润函数。

$$\max \quad z' = 2x_1 + 3x_2$$

服从线性约束

$$\begin{aligned} 4x_1 + 2x_2 &\leqslant 8 \\ 2x_1 + 4x_2 &\leqslant 8 \\ x_1,\ x_2 &\geqslant 0 \end{aligned}$$

将该问题转换为相同约束下最小化 z 的问题

$$\left.\begin{aligned} \min \quad & z = -2x_1 - 3x_2 \\ \text{s.t.} \quad & 4x_1 + 2x_2 \leqslant 8 \\ & 2x_1 + 4x_2 \leqslant 8 \\ & x_1,\ x_2 \geqslant 0 \end{aligned}\right\} \tag{4.1}$$

容易看出在 x_1-x_2 平面内，满足问题(4.1)所有约束的线性约束点(x_1,x_2)的集合成为图4.1中边界线和内点。

表4.1　生产条件和利润

	A	B	可行的数量
劳动时间	4	2	8
材料	2	4	8
利润/百万美元	2	3	

对于固定的z，满足$z=-2x_1-3x_2$的点集构成一条线。随着z的变化，这条线平行移动。这个例子的最优值就是这条线至少有一点与线性约束集合相交时最小的z值。从图4.1可以看出，该问题的最优解是

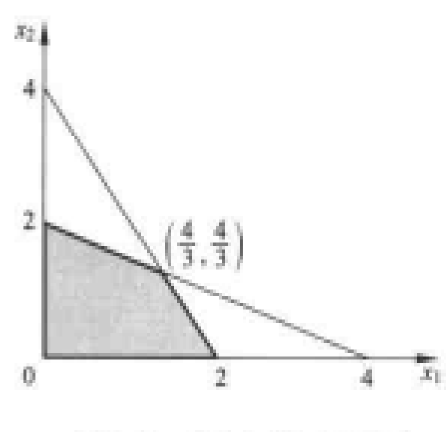


图4.1 例4.1的可行区域

$$x_1=4/3,\quad x_2=4/3,\quad z=-20/3$$

带有模糊资源和目标的线性规划问题可以描述如下：

$$\left.\begin{aligned}&\widetilde{\max}\quad z=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\\&\text{s.t.}\quad \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}\leqslant\tilde{b}_i,\quad i=1,2,\cdots,m\\&\qquad\ \boldsymbol{x}\geqslant 0\end{aligned}\right\}\tag{4.2}$$

其中$\boldsymbol{x}\in\mathbb{R}^n$是决策向量；$\boldsymbol{A}\in\mathbb{R}^{m\times n}$和$\boldsymbol{c}\in\mathbb{R}^n$分别是清晰约束矩阵和目标向量；$\tilde{\boldsymbol{b}}=(\tilde{b}_1,\cdots,\tilde{b}_m)$是$m$维列向量。约束矩阵$\boldsymbol{A}$可以重写为：

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{a}_1^{\mathrm{T}}\\\boldsymbol{a}_2^{\mathrm{T}}\\\vdots\\\boldsymbol{a}_m^{\mathrm{T}}\end{bmatrix}=\begin{bmatrix}a_{11}&a_{12}&\cdots&a_{1n}\\a_{21}&a_{22}&\cdots&a_{2n}\\\vdots&\vdots&&\vdots\\a_{m1}&a_{m2}&\cdots&a_{mn}\end{bmatrix}$$

事先给出模糊目标函数的目标z_0及其对应的容差p_0，模糊约束b_i及其对应的容差p_i。此后对于模糊目标和模糊约束不加以特别区分，它们对应的区域可以用区间$[b_i,b_i+p_i](\forall i)$来表示。于是式(4.2)可以如下考虑：

$$\left.\begin{aligned}&\text{寻找}\quad \boldsymbol{x}\\&\text{s.t.}\quad \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\geqslant z_0\\&\qquad\ \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}\leqslant b_i,\quad i=1,2,\cdots,m\\&\qquad\ \boldsymbol{x}\geqslant \boldsymbol{0}\end{aligned}\right\}\tag{4.3}$$

模糊集理论中，模糊目标函数和模糊约束用它们对应的隶属度函数来定义。为了简单起见，假设模糊目标的隶属度函数μ_0是非降连续线性函数，模糊约束的隶属度函数$\mu_i(\forall i)$是非增连续线性型*隶属度函数*(linear membership function)[①]。目标和约束的隶属度函数如图4.2所示，用公式表示如下：

$$\mu_0(\boldsymbol{x})=\begin{cases}1, & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\geqslant z_0\\1-\dfrac{z_0-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}}{p_0}, & z_0-p_0<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}<z_0\\0, & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\leqslant z_0-p_0\end{cases}\tag{4.4}$$

① 译者注：由式(4.3)可知，优化的1个约束是$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\geqslant z_0$，因此模糊变量$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$越大，则其隶属度越接近1。同理可知，模糊变量$\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}$越接近0，则其隶属度越接近1。

对于第 i 个资源约束，$i=1,2,\cdots,m$

$$\mu_i(x)=\begin{cases}1, & a_i^{\mathrm{T}}x\leqslant b_i\\ 1-\dfrac{a_i^{\mathrm{T}}x-b_i}{p_i}, & b_i<a_i^{\mathrm{T}}x<b_i+p_i\\ 0, & a_i^{\mathrm{T}}x\geqslant b_i+p_i\end{cases}\tag{4.5}$$

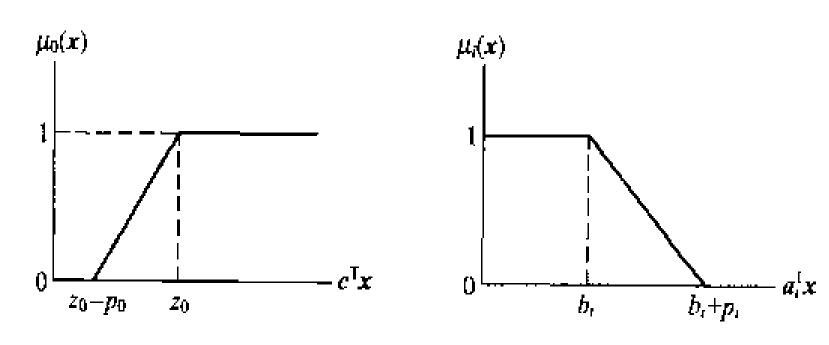


图 4.2　模糊目标约束和第 i 个模糊约束的隶属度函数

Zimmermann 采用 Bellman 和 Zadeh 的最大最小算子来求解式(4.3)～式(4.5)。最优解可以通过求解下式获得：

$$\max_{x\in(\mathbf{R}^n)^+}\mu_{\tilde{S}}(x)=\max\{\min\{\mu_0(x),\mu_i(x)\mid i=1,2,\cdots,m\}\}^{①}\tag{4.6}$$

优化问题(4.6)可以重写为

$$\max_{x\in(\mathbf{R}^n)^+}\mu_{\tilde{S}}(x)=\max\left\{0,\min\left\{1,1-\frac{z_0-c^{\mathrm{T}}x}{p_0},1-\frac{a_i^{\mathrm{T}}x-b_i}{p_i}\middle| i=1,2,\cdots,m\right\}\right\}\tag{4.7}$$

其中 $\mu_{\tilde{S}}$ 是决策空间 $\tilde{S}$ 的隶属度函数，$\mu_{\tilde{S}}=\min\{\mu_0,\mu_i|i=1,2,\cdots,m\}$。如果 $\alpha=\mu_{\tilde{S}}$，则规划对应于：

$$\max\quad \alpha\tag{4.8}$$

$$\text{s.t.}\quad c^{\mathrm{T}}x\geqslant z_0-(1-\alpha)p_0\tag{4.9}$$

$$a_i^{\mathrm{T}}x\leqslant b_i+(1-\alpha)p_i,\quad i=1,2,\cdots,m\tag{4.10}$$

$$x\geqslant \mathbf{0},\quad \alpha\in[0,1]\tag{4.11}$$

一般来说，采用单纯形方法在清晰线性规划中可以获得惟一的最优解(x^*,α^*)。设 x^* 是模糊规划(式(4.2))具有最高隶属度的解。其含义是通过最优解 x^* 达到了目标和约束之间最好的平衡。然而隶属度函数通常不是决策者的偏好函数。有可能决策者并不希望得到惟一的 x^*。另一方面，由于用作计算的原始数据是非精确和模糊的，惟一的精

① 译者注：等号右端的 min 是针对某个解 x 的 $m+1$ 个隶属度函数值，从中选择一个最小的来表征该解 x 的质量。等号右端的 max 是针对解空间中的所有点，从中寻找质量最好的点。

确的最优解在模糊规划中没有意义。汪定伟和唐加福提出了模糊最优的概念来替代惟一最优解[616,649,650]。

例 4.2 A公司希望通过生产产品 P_1 和 P_2 来最大化总利润，这两种产品的生产需要两种不同的材料 M_1 和 M_2。可供支配的材料总数和利润数如表 4.2 所示。公司希望在给定这些材料限制的基础上，知道该生产多少单位的 P_1 产品和 P_2 产品以最大化总利润。

表 4.2 生产条件和利润

	P_1	P_2	可行的数量
M_1	2.5	5	150
M_2	5	2	120
利润/百万美元	1.5	2	

对应的生产计划问题是

$$\begin{aligned} \min \quad & z = -1.5x_1 - x_2 \\ \text{s.t.} \quad & 2.5x_1 + 5x_2 \leqslant 150 \\ & 5x_1 + 2x_2 \leqslant 120 \\ & x_1, x_2 \geqslant 0 \end{aligned}$$

该问题的最优解是

$$x_1 = 15, \quad x_2 = 22.5, \quad z = -45$$

我们不像传统数学规划问题中那样为约束的不等号右边给出严格数字值，而假设决策者具有表 4.3 所示的模糊目标和模糊约束。隶属度函数和隶属度约束可以由下面的公式表示。对于目标函数有

$$\mu_0(\boldsymbol{x}) = \begin{cases} 1, & -1.5x_1 - x_2 \leqslant -47 \\ 1 - \dfrac{-1.5x_1 - x_2 + 47}{5}, & -47 < -1.5x_1 - x_2 < -42 \\ 0, & -1.5x_1 - x_2 \geqslant -42 \end{cases} \tag{4.12}$$

对于第一和第二约束有

$$\mu_{c1}(\boldsymbol{x}) = \begin{cases} 1, & 2.5x_1 + 5x_2 \leqslant 150 \\ 1 - \dfrac{2.5x_1 + 5x_2 - 150}{10}, & 150 < 2.5x_1 + 5x_2 < 160 \\ 0, & 2.5x_1 + 5x_2 \geqslant 160 \end{cases} \tag{4.13}$$

$$\mu_{c2}(\boldsymbol{x}) = \begin{cases} 1, & 5x_1 + 2x_2 \leqslant 120 \\ 1 - \dfrac{5x_1 + 2x_2 - 120}{5}, & 120 < 5x_1 + 2x_2 < 125 \\ 0, & 5x_1 + 2x_2 \geqslant 125 \end{cases} \tag{4.14}$$

表 4.3　非模糊约束与模糊约束

	非模糊	模　糊	
		$\mu=0$	$\mu=1$
目标函数	−45	−42	−47
第一个约束	150	160	150
第二个约束	120	125	120

对于模糊不等式来说，假设线性隶属度函数从 $\mu=0$ 到 $\mu=1$ 变化，用模糊线性规划问题表示的原始问题可以转换为下面对等的传统线性规划问题：

$$\left.\begin{aligned} \max \quad & \alpha \\ \text{s.t.} \quad & 0.3x_1+0.2x_2-\alpha \geqslant 8.4 \\ & 0.25x_1+0.5x_2+\alpha \leqslant 16 \\ & x_1+0.4x_2+\alpha \leqslant 25 \\ & x_1, x_2 \geqslant 0 \end{aligned}\right\} \tag{4.15}$$

用单纯形方法求解上面的线性规划得到的最优解为

$$x_1=15.069, \quad x_2=23.017, \quad \alpha=0.72414$$

表 4.4 中显示了模糊线性规划问题的最优解和非模糊线性规划问题的最优解。从表 4.4很自然得出的结论是，在这个例子中，决策者可以通过一些附加的材料来获得约 1.38%的附加总利润。因此在模糊线性规划中，决策者不再需要像线性规划那样将问题描述为准确和严格的形式，这是模糊线性规划最主要的优点之一。

表 4.4　非模糊线性规划和模糊线性规划问题的解

非模糊	模　糊
$x_1=15$	$x_1=15.069$
$x_2=22.5$	$x_2=23.017$
$z=-45$	$z=-45.621$
约　束	约　束
1：150	1：152.76
2：120	2：121.38

4.2.2　遗传算法方法

用来确定偏好解的交互式决策过程描述如下：

第 1 步：设计若干初步决策方案。

第 2 步：检查这些决策方案是否满足资源约束。如果不满足则调整方案使其满足所有约束。

第 3 步：比较所有决策方案并从中选中最高目标值的最优决策方案。这里采用了沿梯度方向变异的遗传算法来模仿人的决策过程。

遗传算法中的个体在$(\mathbb{R}^n)^+$中随机生成；每个个体根据其目标函数值决定的被选择概率来复制其后代；后代沿其父代的*加权梯度方向*(weighted gradient direction)产生。在若干代以后，大多数个体将接近惟一的最优解 $\boldsymbol{x}^*$。

最优问题是无约束的最大最小问题，但其目标函数不是连续可导的。因此不能用传统优化方法求解。但对于遗传算法来说正好合适。汪定伟和唐加福将模糊最优解的隶属度函数 $\mu_{\tilde{S}}(\boldsymbol{x})$作为遗传算法的适应值函数。对于遗传算法来说，适应值越大则被选中并产生后代的概率就越大。因此带有较高隶属度的个体有较大的机会来复制后代。

汪定伟和唐加福设计了沿加权梯度方向移动的变异算子。个体将由该变异算子引导从而迅速接近模糊最优解集。在得到目标函数和第 i 个约束的权重 w_0 和 $w_i(i=1,2,\cdots,m)$后，$\mu_{\tilde{S}}(\boldsymbol{x})$的加权梯度可以定义如下：

$$\boldsymbol{G}(\boldsymbol{x}) \equiv \nabla\mu_{\tilde{S}}(\boldsymbol{x}) = \frac{w_0\boldsymbol{c}}{p_0} - \sum_{i=1}^{m}\frac{w_i\boldsymbol{a}_i}{p_i} \tag{4.16}$$[①]

设 $\mu_{\min}=\min\{\mu_0,\mu_i\,|\,i=1,2,\cdots,m\}$，于是有

$$w_i = \begin{cases} 1, & \mu_i = \mu_{\min} \\ \dfrac{\sigma}{\mu_i}, & \mu_{\min} < \mu_i < 1 \\ 0, & \mu_i = 1 \end{cases} \tag{4.17}$$

其中 σ 是充分小的正数(通常 $\sigma=10^{-6}$)。设 $\boldsymbol{x}^{k-1}$是个体 $\boldsymbol{x}^k$ 的后代，沿加权梯度方向的变异可以描述如下[199]：

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \theta\boldsymbol{G}(\boldsymbol{x}) \tag{4.18}$$

其中 θ 是 Erlang 分布的随机步长，由随机数发生器产生。

从式(4.17)和式(4.18)表示的权重和变异的公式可以看出，具有最小隶属度函数的约束将获得完全权重 1。变异将把个体引导到可行区域。当 $\mu_{\tilde{S}}(\boldsymbol{x})$大于 0 时，具有最小隶属度函数的目标和约束得到完全权重。加权的梯度方向将改善当前的最小隶属度值[②]，$\mu_{\tilde{S}}(\boldsymbol{x})$将从而得到改善。因此沿着加权梯度方向的变异将引导大多数个体接近精确最优解 $\boldsymbol{x}^*$。需要指出，$\boldsymbol{x}^{k+1}$可能跳出可行区域，但通常可由加权梯度引导回来[③]。随机步长 θ

---

① 译者注：式(4.16)梯度的含义通过式(4.4)～式(4.7)可以看出。它大致反映了使 $\mu_{\tilde{S}}$ 函数值增加的方向。

② 译者注：由式(4.17)可知，w_i 的选取原则是某个隶属度越小，其权重也就越大(最大为 1)。这个原则反映了对最不能得到满足的约束优先进行改善的愿望。

③ 译者注：如果 x^{k+1}跳出了可行区域，由图 4.2 和式(4.5)可以知道，所违背的约束对应的隶属度值将降低为 0，这样它所对应的权重就会增为 1。由式(4.16)可知，此时变异完全向着增加该约束隶属度值的方向进行，其目的是增加满足这个约束的程度，其结果使解回到可行区域。

可以通过随着迭代进程减小 Erlang 分布参数而逐渐降低，因此当算法终止时解可以保持在可行区域。然而需要指出，加权梯度方向通常并不指向精确最优解 x^*。在图4.3所示的问题中，$n=2,m=2$，不同区域中的加权方向用向量表示。这些加权方向并不指向 x^*①。因此遗传算法可以迅速将个体引导到模糊最优解上，但寻找精确最优解则非常缓慢。幸运的是，实践中决策者并非总是希望得到精确最优解 x^*。余下的问题是如何从模糊最优解中选出偏好解。

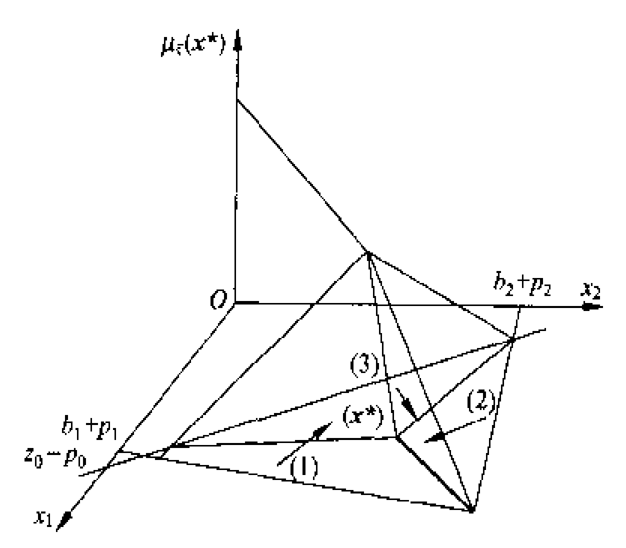


图4.3　不同区域中的加权梯度方向

该算法将真实决策向量用作基因表示。由于实数表示方案不需要转换数字系统，因此在函数优化中得到了成功应用。基因包含 n 个决策变量。设 *pop_size* 表示种群中个体总数。于是对于个体 j，实数向量

$$x(j)=[x_1(j),x_2(j),\cdots,x_n(j)]^{\mathrm{T}},\quad j=1,2,\cdots,pop_size \tag{4.19}$$

就表示染色体。每代中保持种群规模恒定。用 $x^k(j),j=1,2,\cdots,pop_size$ 来表示第 k 代中的第 j 个个体。

4.2.3　交互式方法

对于目标和资源问题来说，目标通常是利润或者生产量，约束通常是不同类型的制造资源。实际工作中，决策者通常对目标值或某些决策变量感兴趣，或者关心资源的消耗。为了从模糊最优解中寻找偏好解，汪定伟设计了一种人机接口方法[616,649,650]。这种方法的基本思路如下。首先，该方法要求决策者给出模糊最优解可接受的隶属度水平 α。其次，通过人机接口引导决策者发现他或她的偏好结构。该方法要求决策者指出哪个判据是他或她最关心的。判据可能是目标、资源或决策变量。随后将模糊最优解 $\tilde{S}_\alpha$ 的 α 截集中具有最大和最小判据值的解在内存中加以保留并每次迭代更新。设遗传算法的代数和种群规模分别是 *max_gen* 和 *pop_size*，算法访问的点数就是 *max_gen*×*pop_size*。这是个很大的数目。因此决策者偏好的解很可能被发现。由于实际情况中决策者一般关注多于一条判据，该方法找到多于一个解。当迭代终止时，这些解及其判据值将通过人机接口

① 译者注：在方向(1)所示的区域里，(x_1,x_2) 接近约束1的边界，其隶属度较小，权重较大，加权梯度方向为指向隶属度函数增加的方向。在方向(2)所在的区域里也是如此。在方向(3)所在的区域里，(x_1,x_2) 接近目标边界，其隶属度较小，权重较大，加权梯度方向为指向目标函数增加的方向。但由式(4.16)知，无论何时，加权梯度都是若干方向的迭加，因此加权梯度方向并不总指向精确最优解 x^* 的方向，而是指向精确最优解 x^* 所在区域的方向。这也就导致用遗传算法优化该模型时出现广度搜索较快，而深度搜索较慢的现象。

显示给决策者。决策者每次可以从中选出两个他或她偏好的解。计算机显示了这两个解与决策者最关心判据关系的曲线[①]。由此决策者可以确定解之间最优的组合。重复这个过程，决策者可以找到其偏好解。很显然，最终解是隶属度大于 α 的解的凸组合。因此该解在 $\widetilde{S}_\alpha$ 的 α 截集中。汪定伟，唐加福和 Fung 将沿着加权梯度方向变异的遗传算法和选择偏好解的人机接口结合起来，针对线性目标和资源优化问题提出了一种新方法，称作非精确方法。该方法的步骤如下。

第 1 步：输入模糊最优解可接受的隶属度水平 α。

第 2 步：输入关心的判据下标集 $C_I, C_I \subseteq \{0,1,2,\cdots,n,n+1,n+2,\cdots,n+m\}$，其中 0 代表目标函数，$j=1,2,\cdots,n$ 代表决策变量，$j=n+1,n+2,\cdots,n+m$ 代表约束。由下式初始化判据 $r \in C_I$ 最小和最大的值并获得对应的解：

$$\min(r) = M, \quad \max(r) = 0$$

$$z_1(r) = \boldsymbol{x}(1), \quad z_2(r) = \boldsymbol{x}(1)$$

其中 M 是充分大的正数（通常 $M=10^6$），$\min(r)$ 和 $\max(r)$ 是判据 r 最小和最大的初始值，$z_1(r)$ 和 $z_2(r)$ 分别是与判据 r 的最小和最大值对应的解。

第 3 步：搜索决策变量 x_i 的上界，用 $x_i^{\mathrm{up}}(i=1,2,\cdots,n)$ 表示。对于 $A \geqslant 0$（这在我们的问题中比较常见）有

$$x_i^{\mathrm{up}} = \max\left\{\frac{b_j + p_j}{a_{ij}} \middle| j = 1,2,\cdots,m\right\}, \quad i = 1,2,\cdots,n \tag{4.20}$$[②]

第 4 步：通过下式产生初始解并计算初始隶属度函数值：

$$x_i(j) = \xi x_i^{\mathrm{up}}$$[③]

$$\xi \in U(0,1), \quad i = 1,2,\cdots,n, \quad j = 1,2,\cdots,pop_size$$

$$\mu_{\widetilde{S}}(\boldsymbol{x}(j)) = \min\{\mu_0(\boldsymbol{x}), \quad \mu_i(\boldsymbol{x}) \mid i = 1,2,\cdots,m\}$$[④]

其中 $U(0,1)$ 表示从 0 到 1 的均匀分布随机数。

第 5 步：设迭代下标 $k=1$。

第 6 步：对于个体 $j(j=1,2,\cdots,pop_size)$，通过下式计算其适应值 $F(j)$ 和被选择概率：

$$F(j) = \mu_{\widetilde{S}}(\boldsymbol{x}(j)) + \epsilon, \quad P(j) = \frac{F(j)}{\sum_{i=1}^{pop_size} F(i)}$$

其中 ϵ 是较小的正数（通常 $\epsilon=10^{-6}$）。它用来确保分母不为零。

第 7 步：通过比例选择方法，选个体 i 作为后代 $j(j=1,2,\cdots,pop_size)$ 的父代。于是有

---

① 译者注：参考例 4.3。

② 译者注：这是考虑其他决策变量均为 0 得到的某变量的上限，用于个体初始化。

③ 译者注：该式得到的是第 j 个个体第 i 个变量的初始值。

④ 译者注：该式得到的是第 j 个个体的隶属度函数值。

$$x^k(j) = x^{k-1}(i) + \theta G(x(i))$$

其中 $G(x(i))$是加权梯度方向，θ 是由 Erlang 分布随机数发生器产生的步长。

第 8 步：对于新产生的个体 $j(j=1,2,\cdots,pop_size)$，计算 $\mu_{\tilde{S}}(x(j))$。设

$$x^* = \mathrm{argmax}\{\mu_{\tilde{S}}(x(j)) \mid j = 1,2,\cdots,pop_size\}$$①

如果 $\mu_{\tilde{S}}(x(j))>\alpha$，检查判据值 $V_r(x(j))(r\in C_I)$ 是否小于 $\min(r)$或大于 $\max(r)$。然后根据 $x(j)$和 $V_r(x(j))$更新 $\min(r)$，$z_1(r)$或 $\max(r)$，$z_2(r)$。②

第 9 步：$k\leftarrow k+1$，如果 $k<max_gen$，返回第 6 步，其中 max_gen 是预先选出的最大遗传代数，与要求的精度有关。否则进入第 10 步。

第 10 步：通过人机接口显示 x^*，$z_1(r)$，$z_2(r)$，$r\in C_I$③。如果决策者认为 z_1 和 $z_2\in\{x^*\mid z_1(r),z_2(r),r\in C_I\}$是偏好解对，所考虑的判据是 r，则显示 z_1 和 z_2 之间判据 r 的曲线④。然后决策者可以确定 z_1 和 z_2 的偏好组合。继续这个组合确定过程直到发现偏好解为止。

4.2.4 数值例子

汪定伟和唐加福采用下面的例子来测试他们的方法[616,649,650]。

例 4.3　某工厂需要在下个月生产两种产品 A 和 B。单位 A 产品和单位 B 产品的净利润分别为 6000 美元和 4000 美元。生产 A 和 B 需要劳动时间和一种关键材料。每生产 1 单位 A 产品需要劳动时间 2 人月和 4 单位材料；每生产 1 单位 B 产品需要劳动时间 3 人月和 2 单位材料。可以支配的劳动时间为 70 人月，可以支配的材料为 100 单位。除此之外，还由工厂的总经理控制库存的 20 单位材料。决策者希望目标利润达到 200 000 美元但至少不要低于 160 000 美元，同时加班工作的时间和安全库存的消耗都不要过量。这是一个典型的模糊目标和约束优化问题，可以由 Zimmermann 的对称模型描述如下[706,707]：

$$\left.\begin{aligned}
\max\quad & \alpha \\
\text{s.t.}\quad & f(x) = 6x_1 + 4x_2 \geqslant 200 - (1-\alpha)40 \\
& g_1(x) = 2x_1 + 3x_2 \leqslant 70 + (1-\alpha)30 \\
& g_2(x) = 4x_1 + 2x_2 \leqslant 100 + (1-\alpha)20 \\
& x_1, x_2 \geqslant 0,\quad \alpha\in[0,1]
\end{aligned}\right\}\qquad(4.21)$$

① 译者注：该式得到的是某一代中最优秀的个体。

② 译者注：如果某代中得到的个体达到决策者对于隶属度的要求（$>\alpha$），同时其判据值又比决策者关心的历史上保存的最大和最小判据值更大或更小，则更新历史数据。

③ 译者注：此时的 x^* 是整个遗传算法搜索过程中找到的最好个体。$z_1(r)$，$z_2(r)$分别是历史上找到的最小和最大的第 r 个判据所对应的 x 值。

④ 译者注：z_1 和 z_2 是属于不同判据的两个历史上找到的最小或最大值所对应的 x_1 和 x_2。显示的是对 z_1 和 z_2 进行凸组合的系数 λ 和比较判据之间关系的曲线。参考例 4.3。

通过 Zimmermann 的容差方法很容易确定精确最优解是 $x^*=[20,\ 15]^T$，$\mu_S(x^*)=0.5$，最优值是 180。首先让我们分析遗传算法中沿加权梯度方向变异的行为。当种群规模 *pop_size*=1 时，一个随机点在最初 8 次迭代中的移动路径如表 4.5 和图 4.4 所示。

表 4.5 沿加权梯度方向变异的路径

	迭代							
	1	2	3	4	5	6	7	8
x_1	5.885	14.128	17.315	24.004	22.939	21.058	18.207	19.873
x_2	5.901	11.396	13.521	17.980	17.181	16.241	14.815	15.926

从表 4.5 和图 4.4 可以看出，这个随机点可以通过沿加权梯度方向的变异快速接近模糊最优解 $\tilde{S}$。值得提出的是并非所有点都可以接近 $\tilde{S}$。比如图 4.4 中区域 D 中的点就不行。在遗传算法中，只要种群规模足够大，这些点很快就会被掩埋在其他区域里。对于上述目标或资源问题，决策者考虑判据 0，1，2。决策者关心净利润和产品 A 和 B 的生产数量。种群规模为 100，可接受的隶属度水平 $\alpha=0.3$。在 228 代以后，遗传算法的结果如表 4.6 所示。

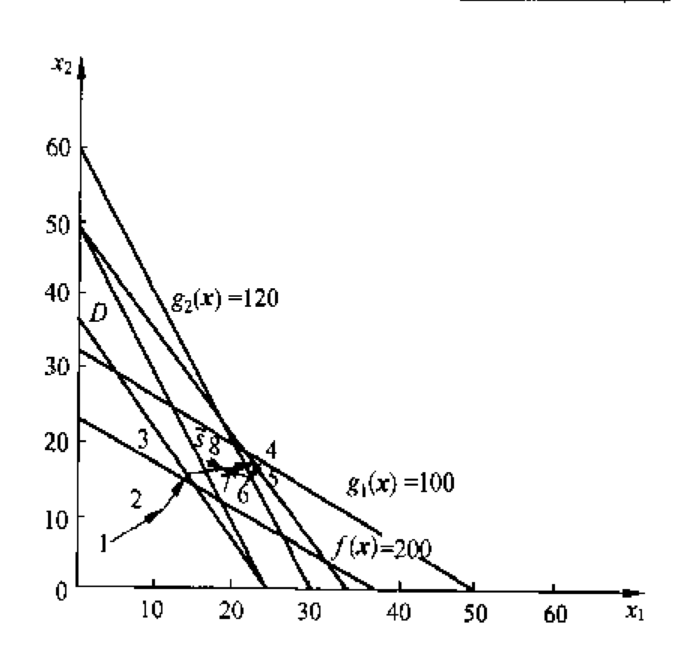


图 4.4 沿着加权梯度方向变异的移动路径

表 4.6 待选择的偏好解

解	x_1	x_2	目标值	$\mu_S(x)$	含 义
x^*	20.0001	14.9997	179.9991	0.49998	精确最优解
$z_1(0)$	19.1533	14.2708	172.0026	0.30007	目标函数最小
$z_2(0)$	20.0293	16.8886	187.7302	0.30528	目标函数最大
$z_1(1)$	15.4291	19.9664	172.4402	0.30809	产品 A 最少
$z_2(1)$	25.6704	5.5495	176.2207	0.31096	产品 A 最多
$z_1(2)$	25.0125	5.4824	172.0044	0.30011	产品 B 最少
$z_2(2)$	15.4291	19.9664	172.4402	0.30809	产品 B 最多

从表4.6可以看出，以这些解作为顶点的多边形几乎覆盖了截集 $\tilde{S}_\alpha$。最优解 x^* 非常接近精确最优解。决策者更看重 $z_2(0)$ 和 $z_2(1)$。这意味着他希望达到较大的利润和生产较多的产品A。他将劳动时间约束选做 $z_2(0)$ 和 $z_2(1)$ 的比较判据[①]。决策者发现如果组合系数 $\lambda=0.9116$，则没有增加劳动时间。于是他选择的解是

$$z=\lambda z_2(1)+(1-\lambda)z_2(0)=[25.1715,6.523]^T$$

对于组合解 z 来说，它的目标函数值、隶属度函数和约束如下：

$$f(z)=177.2383,\qquad \mu_{\tilde{S}}(z)=0.31047$$

$$g_1(z)=70,\qquad g_2(z)=113.7907$$

决策者对此感到满意，于是将这个结果作为下个月的制造计划。

4.3 模糊非线性规划

关于模糊数学规划的研究大部分集中在线性规划[408,519,613,655]、多目标规划[92,553]等领域，对于模糊非线性规划(fuzzy nonlinear programming)(包括模糊二次规划)[616]则很少涉及。然而许多复杂工业系统中的实际问题，存在各种模糊非线性生产计划和调度问题，却无法将其描述成调度模型。因此关于模糊环境下非线性规划(nonlinear programming)建模和优化的研究不仅对于模糊优化理论很重要，而且对于生产计划和调度问题也是颇有价值的。

本节将介绍一种用于解决带有模糊目标和资源约束的模糊非线性规划问题的非精确方法。这种方法针对的是寻找最优解所在的邻域，而不是发现精确最优解。最优解的邻域就是模糊环境下的最优解，邻域中的所有解都是可接受的。本节介绍了一种沿加权梯度方向进行变异的特殊遗传算法，并用来寻找一组具有可接受隶属度的解。这样的隶属度程度能够满足决策者的愿望。随后通过人机接口可以得到不同类型判据下的偏好解。

4.3.1 非线性规划模型

一般来说，带有资源约束的非线性规划问题具有下面的一般形式：

$$\left.\begin{array}{ll}\max & f(\boldsymbol{x})\\ \text{s.t.} & g_i(\boldsymbol{x})\leqslant b_i,\quad i=1,2,\cdots,m\\ & \boldsymbol{x}\geqslant \boldsymbol{0}\end{array}\right\}\qquad(4.22)$$

① 译者注：在例4.3里，决策者优化时考虑的判据是目标函数值(即利润值)、产品A和B的产量、劳动时间和材料消耗。而决策者在进行最终决策选择时考虑的比较判据是劳动时间。

其中 b_i 是清晰的可使用资源。

然而在实际的生产计划问题中，一个周期中可使用资源的数量是不确定的，并带有不同程度的模糊性。同时，决策者定义的目标也是不清楚的目标，或者由于对问题或实际最大化和最小化等方面的限制缺乏全面了解，而在模糊函数中存在某些模糊参数。我们讨论的就是具有下列类型模糊目标和模糊资源约束的非线性规划问题。

1. 决策者期望的目标值并不是实际上的最大化而是一个模糊值。决策者希望达到一个水平 z_0，而且不小于下限 z_0-p_0。同时，随着目标值的增长，决策者满意程度也随之增长。

2. 如果必要，决策者可以少许增加某种资源可使用的数量，比如加班、将库存材料投入使用等。假设可增加资源类型 i 的可使用数量为 $b_i(i=1,2,\cdots,m_1)$，决策者可接受的最大增加量是 p_i，可使用的模糊数量用 $\tilde{b}_i$ 来表示。同时这种模糊资源具有单调非增的隶属度函数。决策者不希望超过可承受的数量来使用这种资源。

3. 其他类型资源 $i(i=m_1+1,m_1+2,\cdots,m)$可使用的数量是非精确的。假设第 i 种资源可使用的数量 $\tilde{b}_i(i=m_1+1,m_1+2,\cdots,m)$的均值为 b_i，误差分别为 p_i^- 和 p_i^+，具有 L-R(左-右)型隶属度函数。决策者希望这种类型的资源用得越充分越好。

进一步假设目标函数 $f(\boldsymbol{x})$和资源约束函数 $g_i(\boldsymbol{x})$在$(\mathbb{R}^n)^+$ 上是非线性、连续并且可导的。

问题是如何确定一种合理的计划使其最优化目标，或者在前述模糊目标和资源约束环境下决策者对于其偏好的判据最“满意”。这种类型的问题属于模糊最优化问题，具有下面一般的形式：

$$\left.\begin{aligned}\widetilde{\max}\quad & f(\boldsymbol{x})\\ \text{s.t.}\quad & g_i(\boldsymbol{x})\leqslant\tilde{b}_i,\quad i=1,2,\cdots,m_1\\ & g_i(\boldsymbol{x})=\tilde{b}_i,\quad i=m_1+1,m_1+2,\cdots,m\\ & \boldsymbol{x}\geqslant\boldsymbol{0}\end{aligned}\right\}\tag{4.23}$$

其中 $\boldsymbol{x}$ 是 n 维决策向量，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$，$\tilde{\boldsymbol{b}}$ 是模糊可使用资源向量，$\tilde{\boldsymbol{b}}=[\tilde{b}_1,\tilde{b}_2,\cdots,\tilde{b}_m]^{\mathrm{T}}$。

汪定伟和唐加福介绍了下面类型的隶属度函数来描述模糊数、模糊目标和模糊约束[616,649,650]。对于资源 $i(i=1,2,\cdots,m_1)$，设 $\mu_{\tilde{b}_i}(\boldsymbol{x})$表示模糊可使用资源 $\tilde{\boldsymbol{b}}$ 的隶属度函数，其定义如下：

$$\mu_{\tilde{b}_i}(\boldsymbol{x})=\begin{cases}1, & g_i(\boldsymbol{x})\leqslant b_i\\ 1-\left(\dfrac{g_i(\boldsymbol{x})-b_i}{p_i}\right)^r, & b_i<g_i(\boldsymbol{x})<b_i+p_i\\ 0, & g_i(\boldsymbol{x})\geqslant b_i+p_i\end{cases}\tag{4.24}$$

其中 $r>0$。$\mu_{\tilde{b}_i}(\boldsymbol{x})$是单调非增函数模糊可用资源 $\tilde{\boldsymbol{b}}$ 的使用程度。

对于资源 $i(i=m_1+1,m_1+2,\cdots,m)$，设 $\mu_{\tilde{b}_i}(\boldsymbol{x})$定义如下：

$$\mu_{\tilde{b}_i}(\boldsymbol{x})=\begin{cases}0, & g_i(\boldsymbol{x})\leqslant b_i-p_i^- \\ 1-\left(\dfrac{b_i-g_i(\boldsymbol{x})}{p_i^-}\right)^r, & b_i-p_i^-<g_i(\boldsymbol{x})\leqslant b_i \\ 1-\left(\dfrac{g_i(\boldsymbol{x})-b_i}{p_i^+}\right)^r, & b_i<g_i(\boldsymbol{x})<b_i+p_i^+ \\ 0, & g_i(\boldsymbol{x})\geqslant b_i+p_i^+\end{cases}\tag{4.25}$$

$\mu_{\tilde{b}_i}(\boldsymbol{x})$是 L-R 型函数，表示对于模糊可用资源 $\tilde{\boldsymbol{b}}$ 估计的准确水平。类似地，设 $\mu_i(\boldsymbol{x})$为第 i 个模糊约束的隶属度函数，定义如下：

$$\begin{aligned}\mu_i(\boldsymbol{x})&=\max_{y\geqslant g_i(x)}\{\mu_{\tilde{b}_i}(y)\}=\mu_{\tilde{b}_i}(g_i(\boldsymbol{x})),\quad i=1,2,\cdots,m_1\\&=\mu_{\tilde{b}_i}(g_i(\boldsymbol{x})),\quad i=m_1+1,m_1+2,\cdots,m\end{aligned}$$

$\mu_i(\boldsymbol{x})$反映了决策者对于第 i 个模糊约束在点 $\boldsymbol{x}$ 的满意程度。

设 $\mu_0(\boldsymbol{x})$描述模糊目标$\widetilde{\max}f(\boldsymbol{x})$，定义如下：

$$\mu_{\tilde{b}_i}(\boldsymbol{x})=\begin{cases}0, & g(\boldsymbol{x})\leqslant z_0-p_0 \\ 1-\left(\dfrac{z_0-f(\boldsymbol{x})}{p_0}\right)^r, & z_0-p_0<f(\boldsymbol{x})<z_0 \\ 1, & f(\boldsymbol{x})\geqslant z_0\end{cases}\tag{4.26}$$

$\mu_0(\boldsymbol{x})$是单调非降连续函数，表示决策者对于模糊目标函数在点 $\boldsymbol{x}$ 的满意程度。

显然，用于描述模糊目标和模糊资源约束的隶属度函数可以由决策者指定为其他类型，比如指数、对数等。

这种问题一个合理的解一般将使决策者对于模糊目标和模糊约束满意。于是就有对模糊目标和模糊约束在满意程度上的平衡问题。处理这种折中的一种常用方法就是在最大程度上使决策者在最关心的判据上满意，其余判据则为不同的满意程度。最关心的判据需要最大化，而其余的判据则保持在一定满意程度之上。于是模糊目标和资源非线性规划(FO/RNP)可以用下面 3 种类型的模型来描述。

1. 最大化最小满意程度(模型 FO/RNP-1)：

$$\left.\begin{aligned}\max\quad & \alpha\\ \text{s.t.}\quad & \mu_0(\boldsymbol{x})\geqslant\alpha\\ & \mu_i(\boldsymbol{x})\geqslant\alpha,\quad i=1,2,3,\cdots,m\\ & \boldsymbol{x}\geqslant\boldsymbol{0}\end{aligned}\right\}\tag{4.27}$$

2. 最大化满意程度的加权和(模型 FO/RNP-2)：

$$\left.\begin{array}{ll}\max & \sum_{i=0}^{m}\beta_i\mu_i(\boldsymbol{x}) \\ \text{s.t.} & \mu_0(\boldsymbol{x}) \geqslant \alpha_0 \\ & \mu_i(\boldsymbol{x}) \geqslant \alpha_0, \quad i=1,2,3,\cdots,m \\ & \boldsymbol{x} \geqslant \mathbf{0}\end{array}\right\} \tag{4.28}$$

其中 α_0 是决策者事先确定的可接受满意程度，$\beta_i(i=0,1,2,\cdots,m)$ 分别是目标和资源约束的权重。它们反映了目标和资源的相对重要性。

3. 最大化决策目标(模型 FO/RNP-3)：

$$\left.\begin{array}{ll}\max & \text{target} \\ \text{s.t.} & \mu_0(\boldsymbol{x}) \geqslant \alpha_0 \\ & \mu_i(\boldsymbol{x}) \geqslant \alpha_0, \quad i=1,2,3,\cdots,m \\ & \boldsymbol{x} \geqslant \mathbf{0}\end{array}\right\} \tag{4.29}$$

其中目标可能是目标函数、决策变量、资源约束等，α_0 是决策者偏好的可接受满意程度。

于是根据不同类型的判据，FO/RNP 的解可以转换为对 FO/RNP-1，FO/RNP-2 和 FO/RNP-3 的解。

定义 4.1 FO/RP 的模糊最优解集是由下式定义的模糊集合 $\tilde{S}$

$$\tilde{S} = \{(\boldsymbol{x},\mu_{\tilde{S}}(\boldsymbol{x})) \mid \boldsymbol{x} \in (\mathbf{R}^n)^+, \mu_{\tilde{S}}(\boldsymbol{x}) = \min\{\mu_0(\boldsymbol{x}),\mu_i(\boldsymbol{x}) \mid i=1,2,\cdots,m\}\} \tag{4.30}$$

设 $S_\alpha=\{\boldsymbol{x}\in(\mathbf{R}^n)^+ \mid \mu_{\tilde{S}}(\boldsymbol{x})\geqslant\alpha\}, \alpha\in[0,1]$。于是 S_α 就是模糊集合 $\tilde{S}$ 的 α 水平截集(α-level cut set)。

定义 4.2 如果对于 $0\leqslant\alpha\leqslant\alpha^*$ 有 S_α 非空，而对于 $\forall\alpha>\alpha^*$ 有 S_α 为空集，则称 α^* 为最佳平衡度。

一般来说，模型 FO/RNP-1 可以找到惟一的最优解 α^*。相应的解 $\boldsymbol{x}^*$(通常并不惟一)就是 FO/RNP 中具有最高隶属度的解。这意味着目标和约束之间的最佳平衡可以在 $\boldsymbol{x}^*$ 点达到。然而 $\boldsymbol{x}^*$ 点不一定是决策者在某些类型判据下最想要的解。另外，在模糊环境下对于决策者来说精确解没有意义。决策者期望的解实际上是满足决策者所偏好的不同判据下目标和资源约束的多个解。基于模糊最优解概念的非精确方法用于寻找最优解所在的邻域，从而使得所有在该领域中的解 $\boldsymbol{x}$ 都是决策者在模糊环境下期望的"最优"解。

下面的小节将介绍用基于遗传算法的非精确方法来求解模型 FO/RNP-1，以及通过人机接口求解 FO/RNP 的整体过程。这个过程为 FO/RNP 模型的实际应用提供了初步的框架。

4.3.2 用于求解 FO/RNP-1 的非精确方法

显然，FO/RNP-1 等价于下面的优化问题 P：

$$\max_{\boldsymbol{x}\in(\mathbf{R}^n)^+} \mu_{\tilde{S}}(\boldsymbol{x}) = \max\min\{\mu_0(\boldsymbol{x}),\mu_1(\boldsymbol{x}),\cdots,\mu_m(\boldsymbol{x})\} \tag{4.31}$$

P 是无约束优化问题，但它的目标不是连续可导的。这个问题不能采用传统优化方法求

解，但可以用遗传算法进行优化[303,455]。本节中讨论由汪定伟和唐加福提出的一种沿着加权梯度方向进行变异的特殊遗传算法[616,649,650]。该算法将变异作为主要算子；仅在后期采用算术组合杂交算子。该算法的基本思想如下。首先，随机产生有 pop_size 个个体的初始种群，个体被选择并产生后代，后代在模糊目标和约束上的隶属度得到增加。随着遗传算法的进行，隶属度小于 α_0（可接受隶属度）的个体比其他个体产生后代的机会少。随着遗传代数的增加，隶属度小于 α_0 的个体最终死去，其他个体得以生存。经过若干代以后，所有个体的隶属度都大于 α_0，即 $S_{\alpha_k}\subseteq S_{\alpha_0}$，大多数个体将会接近最优解。

对于个体 $\boldsymbol{x}$，设 $\mu_{\min}(\boldsymbol{x})=\min\{\mu_0(\boldsymbol{x}),\mu_1(\boldsymbol{x}),\cdots,\mu_m(\boldsymbol{x})\}$。如果 $\mu_{\min}(\boldsymbol{x})\leqslant\mu_0(\boldsymbol{x})<1$，则沿着 $\mu_0(\boldsymbol{x})$ 的梯度方向移动。这样可能改善 $\mu_0(\boldsymbol{x})$ 的值。$\mu_0(\boldsymbol{x})$ 值越小，就可能得到越大的改善。类似地，如果 $\mu_{\min}(\boldsymbol{x})\leqslant\mu_i(\boldsymbol{x})<1$，沿着梯度方向移动 $\mu_i(\boldsymbol{x})$，这样可能改善 $\mu_i(\boldsymbol{x})$ 的值。$\mu_i(\boldsymbol{x})$ 值越小，就可能得到越大的改善。基于上面的思想，构造的梯度方向如下：

$$\boldsymbol{D}(\boldsymbol{x}) = w_0\nabla\mu_0(\boldsymbol{x})+\sum_{i=1}^{m}w_i\nabla\mu_i(\boldsymbol{x}) \tag{4.32}$$

其中对于 $0<\mu_0(\boldsymbol{x})\leqslant 1,\ 0<\mu_i(\boldsymbol{x})\leqslant 1$

$$\begin{cases}\nabla\mu_0(\boldsymbol{x})=r(z_0-f(\boldsymbol{x}))^{r-1}\dfrac{\nabla f(\boldsymbol{x})}{p_0^r}\\ \nabla\mu_i(\boldsymbol{x})=-r(g_i(\boldsymbol{x})-b_i)^{r-1}\dfrac{\nabla g_i(\boldsymbol{x})}{p_i^r}, \quad i=1,2,\cdots,m_1\\ \nabla\mu_i(\boldsymbol{x})=-r(g_i(\boldsymbol{x})-b_i)^{r-1}\mathrm{sgn}\{g_i(\boldsymbol{x})-b_i\}\dfrac{\nabla g_i(\boldsymbol{x})}{q_i^r}, \quad i=m_1,m_2,\cdots,m\end{cases} \tag{4.33}$$

对于 $\mu_0(\boldsymbol{x})=0$，则 $\nabla\mu_0(\boldsymbol{x})=\nabla f(\boldsymbol{x})$；对于 $\mu_i(\boldsymbol{x})=0$，则 $\nabla\mu_i(\boldsymbol{x})=\nabla g_i(\boldsymbol{x})$。①

$$q_i=\begin{cases}p_i^-, & g_i(\boldsymbol{x})<b_i\\ p_i^+, & g_i(\boldsymbol{x})\geqslant b_i\end{cases} \tag{4.34}$$

$\boldsymbol{D}(\boldsymbol{x})$ 称作 $\mu_{\tilde{S}}(\boldsymbol{x})$ 的**加权梯度方向**（weighted gradient direction）。$\mathrm{sgn}(x)$ 是符号函数。w_i 是梯度方向权重，定义如下：

$$w_i=\begin{cases}0, & \mu_i=1\\ \dfrac{1}{\mu_i-\mu_{\min}+e}, & \mu_{\min}\leqslant\mu_i<1\end{cases} \tag{4.35}$$

其中 e 是一个充分小的正数，$1/e$ 是最大的权重②。

由 $\boldsymbol{x}_j^k$ 通过沿着加权梯度方向 $D(\boldsymbol{x})$ 产生的子代 $\boldsymbol{x}_j^{k+1}$ 可以描述如下：

---

① 这样选择梯度的目的使得变异算子操作的个体能够沿着隶属度值增加的方向变化。

② 译者注：根据式(4.31)，对于一个个体 $\boldsymbol{x}$ 来说，它的适应值就是它目标函数和约束的隶属度中最小的数值。优化的目标就是使最小的隶属度值增大。于是应该优先改善具有最小隶属度值的目标或者约束。这就是式(4.35)对小隶属度值给以大权重，而将隶属度值为1的目标或约束的权重定为0的原因。

$$x_j^{k+1} = x_i^k + \beta^k D(x_i^k) \tag{4.36}$$

其中 β^k 是具有下降均值的 Erlang 分布的随机步长。Erlang 分布由采用随机数发生器产生。

隶属度 $\mu_{\tilde{S}}(x_j)$ 的计算如下：

$$\mu_{\tilde{S}}(x_j) = \begin{cases} \mu_{\min}(x_j), & \mu_{\min} \geq \alpha_0 \\ \varepsilon\mu_{\min}(x_j), & \mu_{\min} < \alpha_0 \end{cases} \tag{4.37}$$

其中 α_0 是决策者偏好的可接受的满意程度，$\varepsilon \in U(0,1)$。

从式(4.32)～式(4.36)权重和变异的公式可以知道，无法满足的约束具有最小的隶属度 0，会得到最大的权重 $1/e$，因此变异会把个体引导到可行的区域。当 $\mu_{\tilde{S}}(x)>0$ 时，具有最小隶属度的目标或约束得到最大的权重。于是加权梯度方向就会改善最小的隶属度值，从而导致 $\mu_{\tilde{S}}(x)$ 的改善。从式(4.37)可以发现 $x_j \notin S_{\alpha_0}$，我们给它较低的隶属度(但不是 0)，于是它们具有较小的机会被选中产生后代。因此加权梯度方向将引导所有个体接近精确最优解，这些个体构成一个包括精确最优解的邻域。

在汪定伟和唐加福的算法中，沿着加权梯度方向的变异是主算子，算术组合杂交仅在后期使用。该算法采用了比例选择策略在每代中选出新的种群。

4.3.3 交互式方法

汪定伟和唐加福设计了交互式过程来帮助决策者在不同的判据下，从模糊最优解(即最优解的邻域)中选择一个解。该方法描述如下[616,649,650]。首先，要求决策者给出模糊最优解可接受的隶属度水平 α_0。其次，通过人机接口，描述目标和资源约束的隶属度显示在计算机上，以帮助决策者选择隶属度类型。然后引导决策者指出他或她偏好的结构以及哪个判据是他或她最关心的。判据可能包括目标和资源约束最佳的平衡度(FO/RNP-1)，加权满意度之和(FO/RNP-2)和目标、决策变量或资源(FO/RNP-3)。通过沿着加权梯度方向变异的遗传算法可以找到模糊最优解，在模糊最优解的 α 水平截集中的解和具有最大和最小判据值的解在每代中都进行更新。当迭代终止时，解及其判据值会在计算机中显示出来，以供决策者选择。该方法的过程描述如下。

FO/RNP 的整体过程

第 1 步：初始化

(1.1) 输入可接受的满意度 α_0、最大遗传代数 *max_gen* 和种群规模 *pop_size*。

(1.2) 输入决策者考虑的判据集 $C_I=\{0,1,2,\cdots,n,n+1,\cdots,n+m,n+m+1\}$。其中 0 代表目标函数，$j=1,2,\cdots,n$ 代表决策变量，$j=n+1,n+2,\cdots,n+m$ 代表约束，$j=n+m+1$ 代表加权满意度之和。给出判据 $r\in C_I$ 的初值和初始上下界。

(1.3) 输入描述模糊目标和模糊约束的隶属度函数类型。

(1.4) 输入目标和资源约束的权重 β_i。

第2步：通过 $x_i(j)=\xi x_i^{up}$ 随机产生初始种群(其中 $\xi\in U(0,1)$，$i=1,2,\cdots,n$，$j=1,2,\cdots,pop_size$，x_i^{up} 是变量 $\boldsymbol{x}$ 第 i 个元素的上界)。用式(4.37)计算其隶属度 $\mu_{\tilde{S}}(\boldsymbol{x}_j)$。

第3步：设迭代下标 $k=1$。

第4步：对于个体 $j(j=1,2,\cdots,pop_size)$，通过下式计算其适应值 $F(j)$ 和被选择概率 $P(j)$：

$$F(j)=\mu_{\tilde{S}}(\boldsymbol{x}_j)+e,\quad P(j)=\frac{F(j)}{\sum_{i=1}^{pop_size}F(i)}$$

第5步：用父代 $\boldsymbol{x}_i$ 通过式 $\boldsymbol{x}_j^k=\boldsymbol{x}_i^{k-1}+\beta^k\boldsymbol{D}(\boldsymbol{x}_i^{k-1})$ 产生新个体 $\boldsymbol{x}_j$，其中 β^k 是用随机数发生器产生的具有下降均值的 Erlang 分布的随机步长，$D(\boldsymbol{x})$ 用式(4.32)～式(4.35)定义。

第6步：对于个体 j，用式(4.37)计算其隶属度函数 $\mu_{\tilde{S}}(\boldsymbol{x}_j)$，更新最优隶属度 $\mu_{\max}$ 和判据 r 的上下界。

第7步：设 $k+1\to k$。如果 $k\leqslant max_gen$，返回第4步；否则进入第8步。

第8步：输出最优隶属度 $\mu_{\max}$ 和决策者偏好的判据上下界，终止算法。

4.3.4 数值例子

汪定伟和唐加福采用下面的例子来测试他们的方法[616,649,650]。

例4.4　某制造工厂需要在一定的周期内(比如1个月)生产两种类型的产品A和B。生产A和B需要3种类型的资源：R_1，R_2，和 R_3。每生产1单位产品A分别需要2，4，3个单位的资源；每生产1单位产品B分别需要3，2，2个单位的资源。资源 R_1 和 R_2 分别有50单位和44单位。同时总经理还分别管理着另外30单位和20单位这两种材料的安全库存。资源 R_3 估计需要36单位，估计的误差是5单位。假设A和B产品分别计划生产 x_1 单位和 x_2 单位。进一步假设A产品和B产品的单位费用和销售价格分别为 $UC_1=c_1$，$UC_2=c_2$，$US_1=k_1/x_1^{1/a_1}$，$US_2=k_2/x_2^{1/a_2}$[650]。于是决策者希望总利润达到期望水平 z_0，起码不小于下界 z_0-p_0。这是典型的FO/RNP问题，它可以描述如下：

$$\begin{aligned}
\max\quad & \tilde{f}(\boldsymbol{x})=k_1x_1^{-1/a_1}-c_1x_1+k_2x_2^{-1/a_2}-c_2x_2\\
\text{s.t.}\quad & 2x_1+3x_2\leqslant\widetilde{50},\quad 4x_1+2x_2\leqslant\widetilde{44}\\
& 3x_1+2x_2=\widetilde{36},\quad x_1,x_2\geqslant 0\\
& p_1=30,\quad p_2=20,\quad p_3^-=p_3^+=5.0,\quad r=1\\
& k_1=50,\quad c_1=8.0,\quad k_2=45,\quad c_2=10,\quad a_1=a_2=2
\end{aligned}$$

对于上面的FO/RNP问题，决策者考虑净利润(目标)，产品A和B的产量和对资源 R_1，R_2，和 R_3 的消耗数量。设 $pop_size=80$。52代以后在不同判据下的结果如表4.7所

示。每代的最大满意度如表4.8所示。这个结果由汪定伟和唐加福的非精确方法得到，该方法结合了遗传算法和交互式过程。

表 4.7 在不同判据下的结果($z_0=150, z_0-p_0=120, \alpha_0=0.25$)

判据	x_1	x_2	$f(x)$	$\mu_S(x)$	资 源	含 义
α^*	9.76222	5.06271	128.75000	0.29167		精确最优解
0(1)	7.88529	6.00671	127.54300	0.25143		目标函数最小
0(2)	9.76222	5.06271	128.75000	0.29167		目标函数最大
1(1)	7.69458	4.80628	127.73060	0.25769		产品A最少
1(2)	10.64834	3.80453	127.70040	0.25668		产品A最多
2(1)	8.89719	3.76743	127.63340	0.25445		产品B最少
2(2)	8.63742	6.40122	127.68870	0.25629		产品B最多
3(1)	8.89719	3.76743	127.63340	0.25445	29.09666	资源 R_1 最少
3(2)	8.89301	6.31004	127.90030	0.26334	36.71612	资源 R_1 最多
4(1)	7.69458	4.80628	127.73060	0.25769	40.39088	资源 R_2 最少
4(2)	10.64834	3.80453	127.70040	0.25668	50.20244	资源 R_2 最多
5(1)	7.69458	4.80628	127.73060	0.25769	32.69630	资源 R_3 最少
5(2)	9.78006	5.20287	128.74040	0.25082	39.74592	资源 R_3 最多

表 4.8 每次迭代的最大满意度($z_0=150, z_0-p_0=120, \alpha_0=0.25$)

迭代数	个体数	$\mu_S(x)$	迭代数	个体数	$\mu_S(x)$	迭代数	个体数	$\mu_S(x)$
1	66	0.26351	19	51	0.29167	37	67	0.29167
2	66	0.26351	20	51	0.29167	38	80	0.29167
3	66	0.26351	21	51	0.29167	39	80	0.29167
4	28	0.26393	22	51	0.29167	40	80	0.29167
5	28	0.26393	23	68	0.29167	41	78	0.29167
6	76	0.28529	24	68	0.29167	42	78	0.29167
7	76	0.28529	25	68	0.29167	43	80	0.29167
8	17	0.28894	26	75	0.29167	44	80	0.29167
9	17	0.28894	27	76	0.29167	45	80	0.29167
10	17	0.28894	28	71	0.29167	46	80	0.29167
11	47	0.28955	29	71	0.29167	47	80	0.29167
12	30	0.29081	30	76	0.29167	48	80	0.29167
13	30	0.29081	31	72	0.29167	49	80	0.29167
14	48	0.29150	32	67	0.29167	50	80	0.29167
15	48	0.29150	33	68	0.29167	51	80	0.29167
16	7	0.29164	34	74	0.29167	52	80	0.29167
17	76	0.29164	35	78	0.29167			
18	62	0.29167	36	79	0.29167			

从表4.7中决策者可以选出合适的解。比如，他或她可能选择具有最大目标以及在目标和各种资源之间最佳平衡的解。从表4.8可以发现迭代中个体可以迅速接近模糊最优解 $\tilde{S}$。仅经过18代个体就很接近精确最优解(9.76222, 5.06271)，最优的平衡度为$\alpha^*=0.29167$。

4.4 模糊非线性混合整数目标规划

目标规划(goal programming)的思想由Charnes和Cooper在20世纪60年代提出。自此不同领域中的许多研究者对这种方法都有贡献，比如Ijiri[316]、Lee[403]、Ignizio[315]、gen与Ida[222,236]。本节中将介绍应用遗传算法解决串并联系统可靠性优化问题。这种问题可以用带有不精确混合整数信息的模糊非线性混合整数目标规划(fuzzy nonlinear mixed-integer goal programming)(f-nMIGP)问题来表示。解决这种用模糊非线性混合整数目标规划表示的系统可靠性优化问题的主要思想，是将原始问题转换为非线性混合整数规划问题，然后用遗传算法求解该问题。带有模糊目标的可靠性设计问题在5.2节进行介绍，这是一种典型的实际模糊非线性混合整数目标规划问题。这个问题也是一个多目标优化问题。

4.4.1 模糊非线性混合整数目标规划模型

模糊非线性混合整数目标规划(f-nMIGP)的典型表示如下。寻找 $\boldsymbol{x}^i$ 和 $\boldsymbol{x}^r$ 来优化下面的模糊目标并服从 m 个非线性系统约束：

$$
\left.\begin{aligned}
f_k(\boldsymbol{x}^i,\boldsymbol{x}^r) &\lesssim b_k, \quad k=1,2,\cdots,q_0\\
f_k(\boldsymbol{x}^i,\boldsymbol{x}^r) &\gtrsim b_k, \quad k=q_0+1,\cdots,q_1\\
f_k(\boldsymbol{x}^i,\boldsymbol{x}^r) &\eqsim b_k, \quad k=q_1+1,\cdots,q_2
\end{aligned}\right\} \tag{4.38}
$$
$$
\text{s.t.}\quad g_c(\boldsymbol{x}^i,\boldsymbol{x}^r)\leqslant G_c,\quad c=1,2,\cdots,m
$$

其中 $\boldsymbol{x}^i$ 是 n 维整数向量；$\boldsymbol{x}^r$ 是 n 维实数向量；符号$\lesssim$(模糊小于)指的是近似小于或等于期望水平 b_k，这表示对于比 b_k 大的数值在某容差限制之内决策者仍会满意。符号$\gtrsim$(模糊大于)指的是近似大于或等于期望水平 b_k，这表示对于比 b_k 小的数值在某容差限制之内决策者仍会满意。符号$\eqsim$(模糊等于)指的是 $f_k(\boldsymbol{x}^i,\boldsymbol{x}^r)$应该在期望值 b_k 的临近，这表示对于比 b_k 小或比 b_k 大的数值在限制之内决策者仍会满意。$f_k(\boldsymbol{x}^i,\boldsymbol{x}^r)$是第 k 个非线性目标约束，$g_c(\boldsymbol{x}^i,\boldsymbol{x}^r)$是第 c 个非线性系统约束，b_k 是目标 k 的目标值，G_c 是系统约束 c 的可利用资源，q_0 是模糊最小化目标约束的数量，q_1-q_0 是模糊最大化目标约束的数量，q_2-q_1是模糊等于目标约束的数量，m 是系统约束的数量。

模糊目标的隶属度函数可以定义如下：对于模糊等于的情况有

$$\mu_k(f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}}))=\begin{cases}0, & f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})<b_k-t_k^{\mathrm{L}}\\ \dfrac{f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})-(b_k-t_k^{\mathrm{L}})}{t_k^{\mathrm{L}}}, & b_k-t_k^{\mathrm{L}}\leqslant f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})\leqslant b_k\\ 1, & f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})=b_k\\ 1-\dfrac{f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})-b_k}{t_k^{\mathrm{R}}}, & b_k\leqslant f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})\leqslant b_k+t_k^{\mathrm{R}}\\ 0, & f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})>b_k+t_k^{\mathrm{R}}\end{cases} \quad (4.39)$$

对于模糊小于的情况有

$$\mu_k(f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}}))=\begin{cases}1, & f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})<b_k\\ 1-\dfrac{f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})-b_k}{t_k^{\mathrm{R}}}, & b_k\leqslant f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})\leqslant b_k+t_k^{\mathrm{R}}\\ 0, & f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})>b_k+t_k^{\mathrm{R}}\end{cases} \quad (4.40)$$

对于模糊大于的情况有

$$\mu_k(f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}}))=\begin{cases}0, & f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})<b_k-t_k^{\mathrm{L}}\\ \dfrac{f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})-(b_k-t_k^{\mathrm{L}})}{t_k^{\mathrm{L}}}, & b_k-t_k^{\mathrm{L}}\leqslant f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})\leqslant b_k\\ 1, & f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})>b_k\end{cases} \quad (4.41)$$

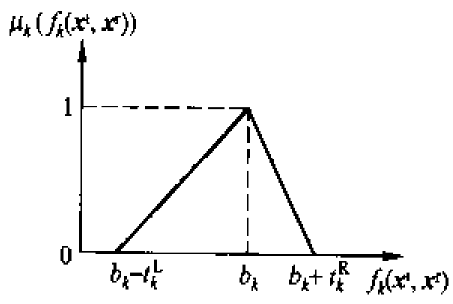


图 4.5 模糊等于的隶属度函数

隶属度函数是严格单调降(或增)函数，并且是关于 $f_k(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})$ 的连续函数，t_k^{L} 是对 b_k 的最大左容差，而 t_k^{R} 是对 b_k 的最大右容差(图 4.5～图 4.7)。然后公式(3.38)就转换为下面的非线性混合整数规划问题(nMIP)：

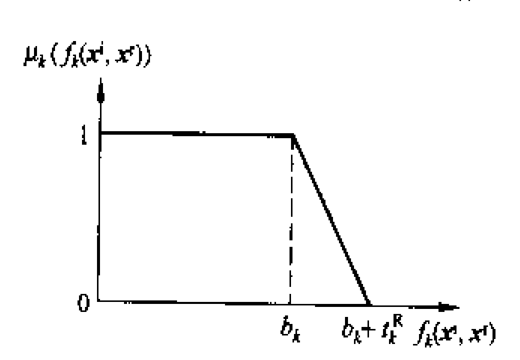


图 4.6 模糊最小化的隶属度函数

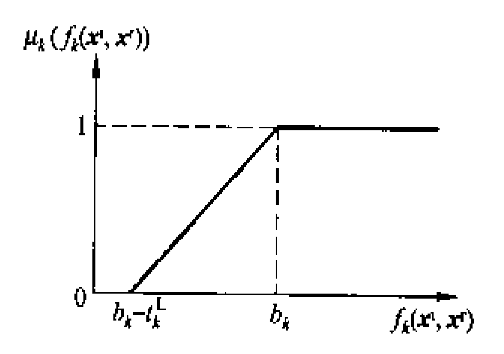


图 4.7 模糊最大化的隶属度函数

$$
\left.
\begin{array}{lll}
\max & \sum_{k=1}^{q_2} w_k \lambda_k & \\
\text{s. t.} & \lambda_k = \mu_k(f_k(\boldsymbol{x}^i, \boldsymbol{x}^r)), & k = 1,2,\cdots,q_2 \\
& g_c(\boldsymbol{x}^i, \boldsymbol{x}^r) \leqslant G_c, & c = 1,2,\cdots,m \\
& 0 \leqslant \lambda_k \leqslant 1, & k = 1,2,\cdots,q_2
\end{array}
\right\} \tag{4.42}
$$

其中 w_k 是由决策者指定的合适的权重因子。决策者需要解决的问题之一就是如何正确设置目标之间的相对重要性。

4.4.2 遗传算法方法

Gen，Ida 和 Kim 提出了一种用于解决表示为 f-nMIGP 模型的串并联系统可靠性优化问题的遗传算法[224,230]。

表示和初始化　设 $\boldsymbol{V}$ 表示种群中的染色体。染色体可以表示如下：

$$\boldsymbol{V} = [(x_1^i, x_1^r), (x_2^i, x_2^r), \cdots, (x_n^i, x_n^r)]$$

这种染色体表示方法对于操作带有 n 维混合整数决策向量的 f-nMIGP 模型比较合适。定义 *pop_size* 表示染色体数量。在定义域中随机产生染色体来进行初始化。

评价　评价函数考虑了权重因子，表示如下：

$$\text{eval}(\boldsymbol{V}_p) = \sum_{k=1}^{q_2} w_k \lambda_k, \quad p = 1,2,\cdots,pop_size$$

其中 λ_k 由式(4.42)给定，w_k 是表示目标(或目标函数)之间相对重要性的权重因子，由决策者指定。采用下面的公式可以在每代中保持具有最大适应值的最佳个体 $\boldsymbol{V}^*$。

$$\boldsymbol{V}^* = \text{argmax}\{\text{eval}(\boldsymbol{V}_p) \mid p = 1,2,\cdots,pop_size\}$$

选择(selection)　选择方法组合了轮盘赌选择和最优性方法。**轮盘赌选择**(roulette wheel selection)根据每个适应值占总适应值的比例来产生新一代，**最优性方法**(elitist method)为下一代保持最优染色体，克服了采样带来的随机误差。选择过程描述如下：

第 1 步：对每个染色体 $\boldsymbol{V}_p(p=1,2,\cdots,pop_size)$ 计算累积概率 a_p。

第 2 步：在[0,1]区间中产生随机数 r。

第 3 步：如果 $r \leqslant a_1$，选择第 1 个染色体 $\boldsymbol{V}_1$；否则选择满足 $a_{p-1} < r \leqslant a_p$ 的第 p 个染色体 $\boldsymbol{V}_p$。

第 4 步：重复第 2 至第 3 步 *pop_size* 次以获得 *pop_size* 个下一代的染色体。

第 5 步：如果下一代中没有选择最佳染色体，在新种群中随机选择一个染色体并被最佳染色体替换。

杂交(crossover)　采用了**算术杂交**(arithmetical crossover)，用两个染色体的凸组合来定义[224]。如果约束集合凸，则算术杂交确保两个可行的父代产生两个可行的子代。

杂交概率 p_c 定义了执行杂交操作的期望染色体数量 $p_c * pop_size$。

由于问题的混合本质，在染色体中存在两种类型的变量：实数变量和整数变量。显然算术杂交对于整数变量将产生实数。我们简单地截去杂交操作产生的实数的小数部分以得到整数部分。

变异 采用了均匀变异(uniform mutation)。变异概率 p_M 定义了执行变异操作的期望染色体数量 $p_M * pop_size$。产生[0,1]区间里的随机数 r，如果 $r<p_M$ 则选择染色体进行变异。对于选出的父代 V，其元素 x_j 随机地选出来进行变异，产生的后代为 $V'=[x_1,\cdots,x_j',\cdots,x_n]$，其中 x_j' 是在 $[x_j^L,x_j^U]$ 中均匀分布的随机值。如果 x_j' 是整数变量，变异算子返回随机整数。该算子确保遗传算法可以在搜索空间中任意搜索，但缺点是在后期比较分散。

柯西方法(Cauchy's method) Gen，Ida 和 Kim 采用了柯西方法(也叫做最速下降法)作为改善遗传算法性能的局部优化方法[224]：

$$\boldsymbol{V}'=\boldsymbol{V}+\alpha\nabla f(\boldsymbol{V})$$

其中正数 α 是步长参数。$\nabla f(\boldsymbol{V})$ 是梯度方向向量，$\boldsymbol{V}'$ 是柯西方法得到的染色体。

参数 α 根据下面的方法进行计算。在不同的情况下存在不同方向的 $\nabla f(\boldsymbol{V})$。如果 $V+\alpha\nabla f(\boldsymbol{V})$ 对于不等式约束不可行，设 α 为 $[0,\alpha]$ 区间内的随机数直到可行为止。如果在事先确定的迭代数之后仍然无法通过上述过程找到可行解，设 $\alpha=0$。对于最大化目标函数选择梯度方向可能较好；对于最小化目标函数选择负梯度方向可能较好。

由于问题存在若干目标函数，采用下面的公式计算梯度方向：

$$\nabla f(\boldsymbol{V})=\sum_{i=1}^{n}w_i\nabla f_i(V)$$

其中 n 是目标函数的数量。

整体过程 对 f-nMIGP 问题的算法如下：

第 1 步：设置参数。设置种群规模(pop_size)，变异率(p_M)，杂交率(p_c)，最大代数(max_gen)和初始代数 $t=0$。

第 2 步：初始化。随机产生初始种群 $\boldsymbol{V}_p(p=1,2,\cdots,pop_size)$。

第 3 步：执行杂交操作。在[0,1]区间中产生随机数 r。如果 $r<P_c$，则选择当前的染色体进行杂交。重复该过程 pop_size 次。对于每个染色体对 $(\boldsymbol{V}_1,\boldsymbol{V}_2)$，杂交操作将产生下面 2 个后代 $(\boldsymbol{V}_1',\boldsymbol{V}_2')$：

$$\boldsymbol{V}_1'=c_1\boldsymbol{V}_1+c_2\boldsymbol{V}_2$$
$$\boldsymbol{V}_2'=c_2\boldsymbol{V}_1+c_1\boldsymbol{V}_2$$

其中 $c_1+c_2=1$，c_1 是[0,1]区间上的随机数。

第 4 步：执行变异操作。在[0,1]区间中产生随机数 r。如果 $r<P_M$，则选择染色体 $\boldsymbol{V}$ 进

行杂交。重复该过程 *pop_size* 次。随机选择给定父代 $\boldsymbol{V}$ 的元素 $\boldsymbol{x}_j$。采用均匀变异方法对 $\boldsymbol{x}_j$ 进行变异。

第5步：计算适应值。设 $t=t+1$，为每个染色体计算 $\text{eval}(\boldsymbol{V}_p)(p=1,2,\cdots,pop_size)$。

第6步：用轮盘赌和最优性选择方法为下一代选出 *pop_size* 个染色体。

第7步：应用柯西方法。采用下面的公式在将后代插入下一代之前将其移动到局部最优点上。

$$\boldsymbol{V}'_p=\boldsymbol{V}_p+\alpha\nabla f(\boldsymbol{V}_p),\quad p=1,2,\cdots,pop_size$$

第8步：执行终止检验。如果 $t<max_gen$，返回第3步操作下一代；如果 $t=max_gen$，终止算法。

4.4.3 数值例子

Gen，Ida 和 Kim 采用下面3个实例来说明他们方法的有效性[224]。

例4.5 问题不仅包括最大化可靠性，还有费用和重量的最小化。问题描述如下：寻找 $\boldsymbol{x}^{\mathrm{i}}$ 和 $\boldsymbol{x}^{\mathrm{r}}$ 来优化下面的模糊目标，即

$$f_1(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})=\prod_{j=1}^{4}\{1-(1-x_j^{\mathrm{r}})x_j^{\mathrm{i}}\}\gtrsim b_1$$

$$f_2(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}})=\sum_{j=1}^{4}C(x_j^{\mathrm{r}})\left[x_j^{\mathrm{i}}+\exp\left(\frac{x_j^{\mathrm{i}}}{4}\right)\right]\lesssim b_2$$

$$f_3(\boldsymbol{x}^{\mathrm{i}})=\sum_{j=1}^{4}d_jx_j^{\mathrm{i}}\exp\left(\frac{x_j^{\mathrm{i}}}{4}\right)\lesssim b_3$$

$$\text{s.t.}\quad g_1(\boldsymbol{x}^{\mathrm{i}})=\sum_{j=1}^{4}v_j(x_j^{\mathrm{i}})^2\leqslant V$$

$$1\leqslant x_j^{\mathrm{i}}\leqslant 10,\quad j=1,\cdots,4$$

$$0.5\leqslant x_j^{\mathrm{r}}\leqslant 1-10^{-6},\quad j=1,\cdots,4$$

其中 x_j^{i} 为正整数，表示子系统 j 冗余的元件数量，x_j^{r} 为实数，表示子系统 j 元件的可靠性，v_j 表示子系统 j 每个元件重量和体积的乘积，d_j 表示子系统 j 每个元件的重量，$F_1=b_1-t_1^{\mathrm{L}}$，$F_2=b_2+t_2^{\mathrm{R}}$，$F_3=b_3+t_3^{\mathrm{R}}$，$C(x_j^{\mathrm{r}})$是子系统 j 具有可靠性 x_j^{r} 的每个元件的费用，其定义如下：

$$C(x_j^{\mathrm{r}})=\alpha_j\left(\frac{-t_0}{\ln(x_j^{\mathrm{r}})}\right)^{\beta_j},\quad j=1,\cdots,4$$

其中 α_j 和 β_j 是用来表征子系统 j 物理特性的常数，t_0 是元件不能失效的运行时间。

Dhingra 对于这个问题的设计数据在表4.9中给出。在运行遗传算法若干次以完成参数调整后（参见图4.8和表4.9），合理的参数设置为：$pop_size=20$，$p_c=0.4$，$p_m=0.1$，$max_gen=2000$。同时设 $t_1^{\mathrm{L}}=0.25$，$t_2^{\mathrm{R}}=300$，$t_3^{\mathrm{R}}=360$。选择目标的相对权重系数分

别为 0.5,0.25,0.25,这样使得第一个目标比其余两个目标的重要性大一倍。不同种群规模(*pop_size*=10,20,30)的进化过程如图 4.9 所示。

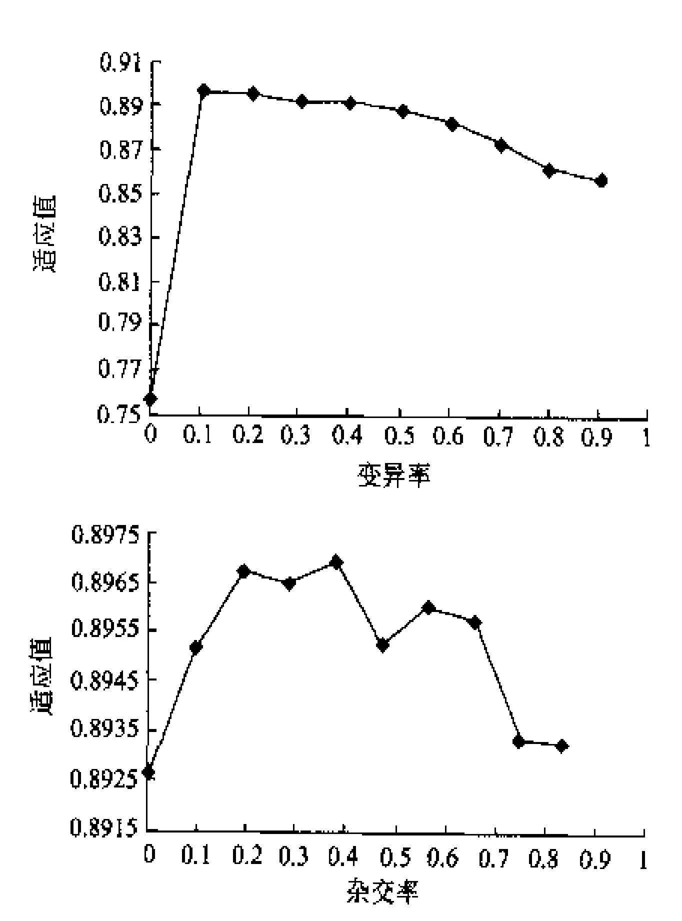


图 4.8 遗传算子的参数

表 4.9 例 4.5 的设计数据

子系统数	4
f_1 的限制	0.75
f_2 的限制	400.0
f_3 的限制	500.0
V 的上限	250.0
运行时间 t_0	1000 小时

子系统	$10^5\times\alpha_j$	β_j	v_j	d_j
1	1.0	1.5	1	6
2	2.3	1.5	2	6
3	0.3	1.5	3	8
4	2.3	1.5	2	7

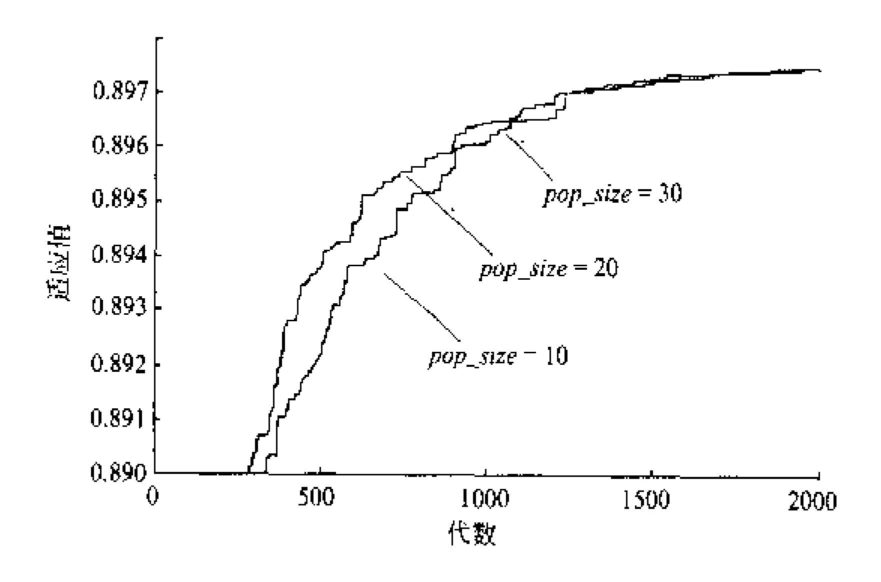


图 4.9　种群规模的选取

Gen 等人采用的方法和 Yokota 等人采用的方法[684]的结果比较如图 4.10 和表 4.10 所示。从比较中可以看出，Gen 等人的方法胜出 Yokota 等人的方法。图 4.10 说明 Gen 等人的方法在收敛到最优妥协解上比 Yokota 等人的方法好。但 Gen 等人方法的平均计算时间要比 Yokota 等人方法的平均计算时间长。

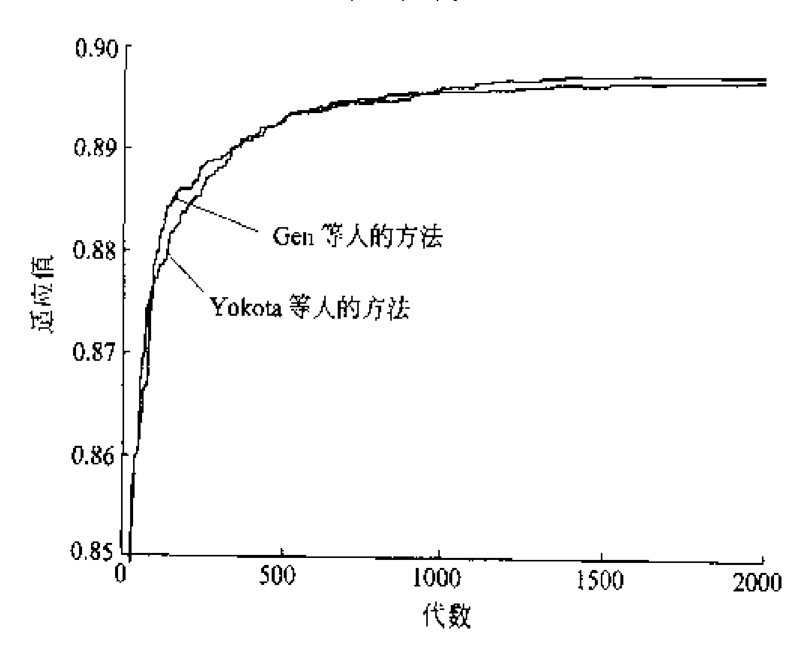


图 4.10　例 4.5 的遗传过程

最优解在第 1923 代找到，其目标函数值为 $f_1(x^i, x^r)=0.982335$，$f_2(x^i, x^r)=135.001259$，$f_3(x^r)=147.048541$，最优解是[(3,0.853070)，(3,0.821421)，(2,0.939903)，(3,0.825620)]。文献[162]给出的最优目标函数值是 $f_1(x^i, x^r)=0.94478$，$f_2(x^i, x^r)=104.472$，$f_3(x^r)=128.727$，最优解是[(2,0.86504)，(3,0.81814)，(2,0.84253)，(3,0.80645)]。

表 4.10　例 4.5 的性能比较

	Yokota 等人的方法	Gen 等人的方法
平均 CPU 时间/s	4.427	18.7125
适应值		
平均	0.896845	0.897456
最差	0.894179	0.896436
最好	0.897654	0.897773
标准偏差	0.000894	0.000339

例 4.6　这是带有 5 个子系统的串并联系统可靠性优化设计问题，是另一个非线性混合整数规划问题实例。该问题可以描述如下：寻找 $\boldsymbol{x}^{\mathrm{I}}$ 和 $\boldsymbol{x}^{\mathrm{r}}$ 来优化下面的模糊目标，即

$$f_1(\boldsymbol{x}^{\mathrm{I}},\boldsymbol{x}^{\mathrm{r}}) = \prod_{j=1}^{5}\{1-(1-x_j^{\mathrm{r}})^{x_j^{\mathrm{I}}}\} \gtrsim b_1$$

$$f_2(\boldsymbol{x}^{\mathrm{I}}) = \sum_{j=1}^{5} d_j x_j^{\mathrm{I}} \exp\left(\frac{x_j^{\mathrm{I}}}{4}\right) \lesssim b_2$$

$$\text{s.t.}\quad g_1(\boldsymbol{x}^{\mathrm{I}}) = \sum_{j=1}^{5} v_j (x_j^{\mathrm{I}})^2 \leqslant G_1$$

$$g_2(\boldsymbol{x}^{\mathrm{I}},\boldsymbol{x}^{\mathrm{r}}) = \sum_{j=1}^{5} C(x_j^{\mathrm{r}})\left[x_j^{\mathrm{I}} + \exp\left(\frac{x_j^{\mathrm{I}}}{4}\right)\right] \leqslant G_2$$

$$x_j^{\mathrm{I}} \geqslant 1,\quad j=1,\cdots,5$$

$$0 \leqslant x_j^{\mathrm{r}} \leqslant 1,\quad j=1,\cdots,5$$

其中 x_j^{I} 为正整数，x_j^{r} 为实数，$C(x_j^{\mathrm{r}})$可以用下面的公式表示：

$$C(x_j^{\mathrm{r}}) = \alpha_j\left(\frac{-t_0}{\ln(x_j^{\mathrm{r}})}\right)^{\beta_j},\quad j=1,\cdots,5$$

该 f-nMIGP 问题的设计数据如表 4.11 所示。参数设置为：$pop_size=20$，$max_gen=2000$，$p_c=0.4$，$p_m=0.1$，$t_1^L=0.25$，$t_2^R=120$。模糊目标的相对权重为 0.75，0.25。结果如表 4.12 和图 4.11 表示。

表 4.11　例 4.6 的设计数据

子系统数	5
f_1 的限制	0.75
f_2 的限制	400.0
g_1 的限制	220.0
g_2 的限制	350.0
运行时间 t_0	1000 小时

续表

子系统	$10^5\times\alpha_j$	β_j	v_j	d_j
1	2.33	1.5	1	7
2	1.45	1.5	2	8
3	0.541	1.5	3	8
4	8.05	1.5	4	6
5	1.95	1.5	2	9

表 4.12 例 4.6 的性能比较

	Yokota 等人的方法	Gen 等人的方法
平均 CPU 时间/s	5.907	19.661
适应值		
平均	0.957442	0.963852
最差	0.946778	0.956742
最好	0.965478	0.968382
标准偏差	0.005282	0.003348

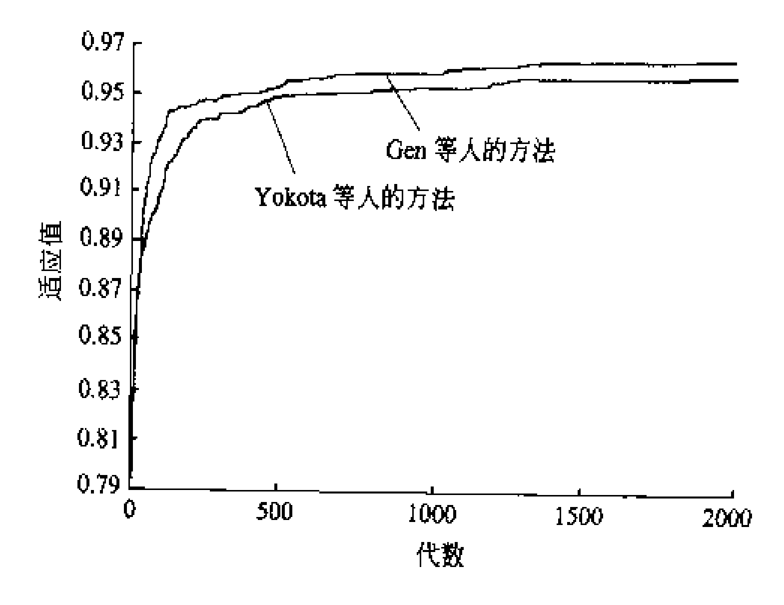


图 4.11 例 4.6 的遗传过程

例 4.7 考虑与例 4.6 类似的 5 个子系统的问题。采用了表 4.13 的设计数据。数学模型如下：寻找 $\boldsymbol{x}^{\mathrm{i}}$ 和 $\boldsymbol{x}^{\mathrm{r}}$ 来优化下面的模糊目标，即

$$f_1(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}}) = \prod_{j=1}^{5}\{1-(1-x_j^{\mathrm{r}})x_j^{\mathrm{i}}\} \gtrsim b_1$$

$$f_2(\boldsymbol{x}^{\mathrm{i}},\boldsymbol{x}^{\mathrm{r}}) = \sum_{j=1}^{5}C(x_j^{\mathrm{r}})\left[x_j^{\mathrm{i}}+\exp\left(\frac{x_j^{\mathrm{i}}}{4}\right)\right] \lesssim b_2$$

$$f_3(\boldsymbol{x}^I) = \sum_{j=1}^{5} d_j x_j^I \exp\left(\frac{x_j^I}{4}\right) \lesssim b_3$$

$$\text{s.t.} \quad g_1(\boldsymbol{x}^I) = \sum_{j=1}^{5} v_j (x_j^I)^2 \leqslant G_1$$

$$x_j^I \geqslant 1, \quad j = 1,\cdots,5$$

$$0 \leqslant x_j^R \leqslant 1, \quad j = 1,\cdots,5$$

其中 x_j^I 为正整数，x_j^R 为实数，$C(x_j^R)$可以用下面的公式表示：

$$C(x_j^R) = \alpha_j \left(\frac{-t_0}{\ln(x_j^R)}\right)^{\beta_j}, \quad j = 1,\cdots,5$$

表 4.13　例 4.7 的设计数据

	子系统数			5
	f_1 的限制			0.75
	f_2 的限制			205.0
	f_3 的限制			240.0
	g_1 的限制			110.0
	运行时间 t_0			1000 小时
子系统	$10^5 \times \alpha_j$	β_j	v_j	d_j
1	2.33	1.5	1	7
2	1.45	1.5	2	8
3	0.541	1.5	3	8
4	8.05	1.5	4	6
5	1.95	1.5	2	9

参数设置为：$pop_size = 20$，$max_gen = 2000$，$p_c = 0.4$，$p_m = 0.1$，$t_1^L = 0.25$，$t_2^R = 40$，$t_3^R = 50$。模糊目标的相对权重为 0.5，0.25，0.25。结果用表 4.14 和图 4.12 表示。从表 4.14 和图 4.12 中可以看出，Gen 等人的方法胜出 Yokota 等人的方法[684]。最优解在第 1914 代找到，其目标函数值为 $f_1(\boldsymbol{x}^I,\boldsymbol{x}^R) = 0.924627$，$f_2(\boldsymbol{x}^I,\boldsymbol{x}^R) = 165.229873$，$f_3(\boldsymbol{x}^I) = 192.481082$，最优解是[(3,0.780943)，(2,0.852900)，(2,0.901031)，(3,0.700890)，(3,0.792657)]。

表 4.14　例 4.7 的性能比较

	Yokota 等人的方法	Gen 等人的方法
平均 CPU 时间/s	5.3005	23.6125
适应值		
平均	0.699208	0.710613
最差	0.654877	0.676798
最好	0.727734	0.730113
标准偏差	0.022590	0.013156

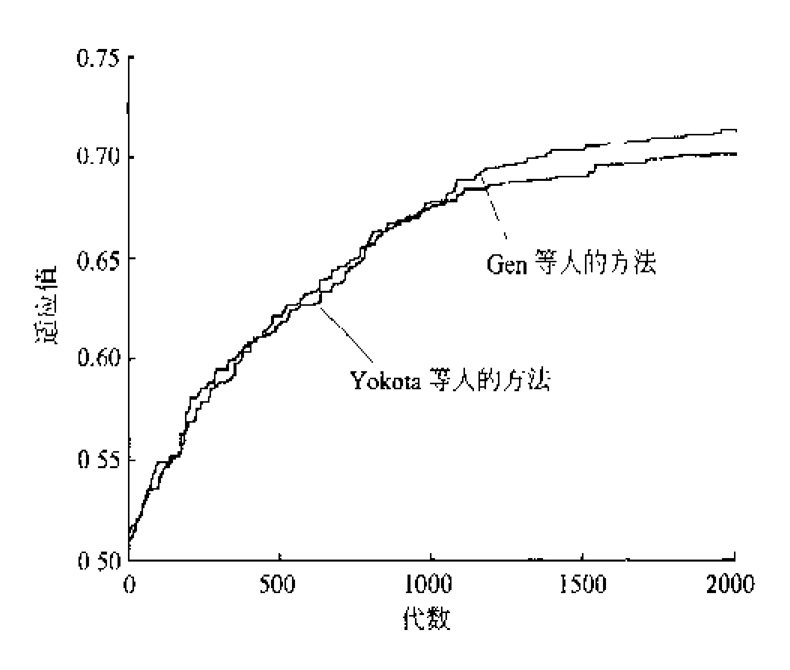


图 4.12　例 4.7 的遗传过程

4.5　模糊多目标整数规划

Sakawa，Shibano 和 Kato 为解决模糊多目标规划问题提出了一种遗传算法[549]。他们将模糊目标引入多目标整数规模问题从而处理决策者判断的不明确性。通过采用参考隶属度的形式来引入决策者的主观偏好，问题可以转换为增广的最小最大问题(augmented minimax problem)。他们采用了一种遗传算法来解决增广的最小最大问题。

4.5.1　问题描述

多目标整数规划问题一般可以描述如下：

$$\left.\begin{array}{ll}\min & z_1(\boldsymbol{x})=\boldsymbol{c}_1\boldsymbol{x}\\ & \cdots\\ \min & z_k(\boldsymbol{x})=\boldsymbol{c}_k\boldsymbol{x}\\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x}\leqslant\boldsymbol{b}\\ & x_j\in\{0,\cdots,v_j\},\quad j=1,\cdots,n\end{array}\right\}\tag{4.43}$$

其中 $\boldsymbol{c}_i=(c_{i1},\cdots,c_{in})$，$\boldsymbol{x}=(x_1,\cdots,x_n)^{\mathrm{T}}$，$\boldsymbol{b}=(b_1,\cdots,b_m)^{\mathrm{T}}$，$\boldsymbol{A}=(a_{ij})$为 $m\times n$ 的矩阵，v_i 是正整数。假设矩阵 $\boldsymbol{A}$ 和向量 $\boldsymbol{b}$ 中每个元素都是正的。一个特殊例子就是多目标多维整数背包问题。然而对于多目标规划问题，当目标函数相互冲突时，一般不存在能够同时最小化所有目标函数的最优解。因此在多目标规划问题中广泛使用 Pareto 最优性的概念来替代最优解的概念。

为了考虑决策者判断的不明确性，采用了针对目标函数的模糊目标，比如“$z_i(\boldsymbol{x})$应该

充分小于或等于某个值”。每个模糊目标的隶属度函数用式(4.44)定义，如图 4.13 所示。

$$\mu_i(z_i(\boldsymbol{x})) = \begin{cases} 0, & z_i(\boldsymbol{x}) > z_i^0 \\ \dfrac{z_i(\boldsymbol{x}) - z_i^0}{z_i^1 - z_i^0}, & z_i^1 < z_i(\boldsymbol{x}) \leqslant z_i^0 \\ 1, & z_i(\boldsymbol{x}) \leqslant z_i^1 \end{cases} \tag{4.44}$$

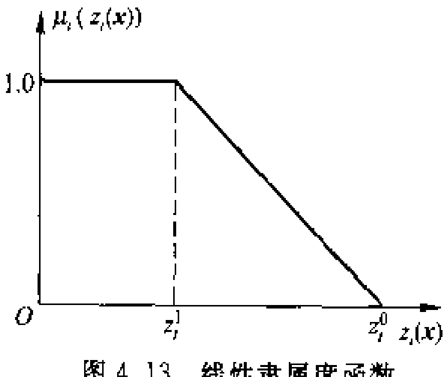


图 4.13 线性隶属度函数

在给定约束下计算每个目标函数的最小值 z_i^{min} 和最大值 z_i^{max} 都进行了计算，目的是帮助决策者确定 z_i^0 和 z_i^1。通过计算出的每个目标函数的最小值和最大值，决策者需要在区间$[z_i^{min}, z_i^{max}]$中来估计 z_i^0 和 $z_i^1 (i=1,\cdots,k)$。Zimmermann 提出了一种确定线性隶属度函数 $\mu_i(z_i(\boldsymbol{x}))$的方法[705]。对每个目标计算最大值和最小值如下：

$$z_i^{min} = \min_{x \in X}\{z_i(\boldsymbol{x}) \mid i = 1,\cdots,k\} \tag{4.45}$$

$$z_i^{max} = \max_{x \in X}\{z_i(\boldsymbol{x}^{i0}),\cdots,z_i(\boldsymbol{x}^{i0}),\cdots,z_i(\boldsymbol{x}^{k0})\} \tag{4.46}$$

然后线性隶属度函数可以通过 $z_i^1 = z_i^{min}$ 和 $z_i^0 = z_i^{max}$ 来确定。

决策者从每个目标函数 $z_i(\boldsymbol{x})$得到线性隶属度函数 $\mu_i(z_i(\boldsymbol{x}))$以后，可以导出一般的联合函数(conjunctive function)：

$$\mu_D(\mu(z_i(\boldsymbol{x}))) = \mu_D(\mu_1(z_1(\boldsymbol{x})),\cdots,\mu_k(z_k(\boldsymbol{x}))) \tag{4.47}$$

则问题就转换为下面的模糊决策问题：

$$\left.\begin{aligned} \max_{x \in X} \quad & \mu_D(\mu(z(\boldsymbol{x}))) \\ \text{s. t.} \quad & \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b} \\ & x_j \in \{0,\cdots,v_j\}, \quad j = 1,\cdots,n \end{aligned}\right\} \tag{4.48}$$ ①

联合函数 $\mu_D(\mu_i(z_i(\boldsymbol{x})))$的值可以理解成决策者对 k 个模糊目标的满意程度。对于联合函数，如果采用 Bellman 和 Zadeh[52]给出的最小算子

$$\min\{\mu_1(z_1(\boldsymbol{x})),\cdots,\mu_k(z_k(\boldsymbol{x}))\}$$

那么模糊多目标整数规划(fuzzy multiobjective integer programming)问题可以转换为

$$\left.\begin{aligned} \min \quad & \max_{i=1,\cdots,k}\{\mu_i - \mu_i(z_i(\boldsymbol{x}))\} \\ \text{s. t.} \quad & \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b} \\ & x_j \in \{0,\cdots,v_j\}, \quad j = 1,\cdots,n \end{aligned}\right\} \tag{4.49}$$ ②

---

① 译者注：$\mu_D(\cdot)$ 的 k 个变量是 x 的 k 目标的隶属度函数值。$\mu_D(\cdot)$ 代表的是决策者对这 k 个目标满意的程度。$\mu_D(\cdot)$ 越大则决策者越满意。故此问题成为最大化问题。

② 译者注：max 的含义是对变量 x，k 个目标与对应的期望存在 k 个差距，找出其中最大的差距。min 的含义是在 x 的可行域中搜索与目标期望最大差异最小的点，即加权意义上的最优点。

4.5.2 增广的最小最大问题

Sakawa, Shibano 和 Kato 引入了一种增广最小最大方法来求解问题(4.49)。设 $\bar{\mu}_i, i=1,\cdots,k$ 表示反映决策者主观上对每个隶属度函数所期望的参考隶属度水平。最优解可以通过求解下面的最小最大问题(minimax problem)得到:

$$\left.\begin{array}{ll}\min & \max\limits_{i=1,\cdots,k}\{\bar{\mu}_i-\mu_i(z_i(\boldsymbol{x}))\} \\ \text{s.t.} & \boldsymbol{Ax}\leqslant \boldsymbol{b} \\ & x_j\in\{0,\cdots,v_j\},\quad j=1,\cdots,n\end{array}\right\}\tag{4.50}$$

如果参考隶属度水平是可达的,则可以找到在参考隶属度水平之上的最优解。如果不可达到参考隶属度水平,则希望找到与参考隶属度水平在最小最大意义上最接近的最优解。

需要指出,最小最大问题的最优解并不惟一。因此需要采用其他信息或判据来从最优解集中选出一个解。Sakawa, Shibano 和 Kato 提出了一种增广最小最大方法来处理这个问题。

$$\left.\begin{array}{ll}\min & \max\limits_{i=1,\cdots,k}\{\bar{\mu}_i-\mu_i(z_i(\boldsymbol{x}))+\rho\sum\limits_{i=1}^{k}(\bar{\mu}_i-\mu_i(z_i(\boldsymbol{x})))\} \\ \text{s.t.} & \boldsymbol{Ax}\leqslant \boldsymbol{b} \\ & x_j\in\{0,\cdots,v_j\},\quad j=1,\cdots,n\end{array}\right\}\tag{4.51}$$ ①

其中 ρ 是充分小的正数。增广部分反映了每个变量到其期望值之间的总偏差。通过求解增广最小最大问题(4.51),可以得到在最小最大意义上带有总最佳满意度的最接近参考隶属度水平的最优解[551,552]。

4.5.3 遗传算法方法

由于增广最小最大问题(4.51)是整数规划问题,其目标和约束都是线性的,而且系数都是正的,Sakawa, Shibano 和 Kato 开发了一种遗传算法来求解该问题。采用双串编码方式来保证任意的染色体都产生可行解。另外还将一种交互式模糊满意方法结合进遗传算法,使得决策者有机会调整隶属度函数。

表示(representation) **双串编码**(double string encoding)方法包括两个部分:上面的部分表示变量的下标,下面的部分表示变量的值。如图 4.14 所示,$s(i)$ 对应着变量 x_i 的下标 i,$x_{s(i)}$ 对应着变量 x_i 的值。

---

① 译者注:正如作者所说的那样,式(4.50)所代表的问题存在多个最优解,即可能存在若干个 $\boldsymbol{x}$,它们的 k 个目标的隶属度值与期望值差距的最大值相同。式(4.51)的意义就是要使得其最优解不仅在 k 个目标的隶属度值与期望值差距的最大值意义上最小化,同时还要使其在所有 k 个目标与其期望值差距之和的意义上最小化。

下标值	$s(1)$	$s(2)$	$\cdots$	$s(i)$	$\cdots$	$s(n)$
	$x_{s(1)}$	$x_{s(2)}$	$\cdots$	$x_{s(i)}$	$\cdots$	$x_{s(n)}$

图 4.14 双串

从给定染色体获得解的直接方法是：

$$x_i = x_{s(i)},\quad i = s(i),\quad i = 1,2,\cdots,n$$

对于图 4.15 给出的例子，解为

$$x_1 = 0,\quad x_2 = 5,\quad x_3 = 2,\quad x_4 = 3,\quad x_5 = 1,\quad x_6 = 2$$

一般地，如果采用给定染色体中的值，问题的约束可能被违反，即通过直接方法获得的解可能是不可行的。因此采用了一种单次通过的过程来进行解码。该方法从左到右扫描染色体，每次固定一个决策变量值 x_i，直到约束被违反为止。由于解码过程是确定性的，而且总是从给定染色体的最左端开始，所以双串编码中的下标字符串的作用就显现出来了。在单字符串编码方法中，染色体中的变量下标是固定的。比如最左边的始终对应 x_1，而最右边的始终对应 x_n。如果对这样的单字符串编码方式采用相同的单次通过过程进行解码，则 x_1 始终优先被固定，然后是 x_2，依次进行下去。于是遗传搜索的范围将被解码方法所限制。在双串编码方法中，变量的下标被首先固定的概率是相同的。整体解码过程如下。

双串解码过程

第 1 步：设 $i=1, \text{sum}_j=0(j=1,\cdots,m)$。

第 2 步：通过下式将带有下标 $s(j)$ 的变量的值固定为 $p_{s(j)}$。

$$p_{s(i)} = \min\left\{\min_{\{j|a_{js(i)}\neq 0\}}\left[\frac{b_j - \text{sum}_j}{a_{js(i)}}\right], x_{s(i)}\right\}$$

其中 $a_{js(i)}$ 是约束的系数。

第 3 步：设 $\text{sum}_j=\text{sum}_j+a_{js(i)}p_{s(i)}, j=1,\cdots,m$，设 $i=i+1$。

第 4 步：如果 $i>n$，终止过程；否则返回第 2 步。

注意，根据上面的解码过程，位置靠左的基因比位置靠右的基因具有优先被固定的机会。为了平衡基因之间的影响，也采用了双环串(ringed double strings)。双串和双环串均显示在图 4.15 中。

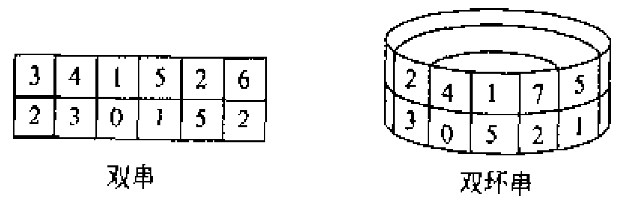


图 4.15 双串和双环串

适应值 每个个体 s 的适应值 $f(s)$ 可以根据下式计算：

$$f(s)=1.0+k\rho-\max_{i=1,\cdots,k}\left\{(\bar{\mu}_i-\mu_i(z_i(x)))+\rho\sum_{i=1}^{k}(\bar{\mu}_i-\mu_i(z_i(x)))\right\}$$

采用线性比例变换方法来调整染色体的适应值，使得最优染色体得到固定数量的期望后代，从而避免复制过多的最优染色体。

复制 Wakawa 和 Shibano 研究了 6 种复制算子的性能：排序选择(ranking selection)、最优性排序选择(elitist ranking)、期望值选择(expected value selection)、最优性期望值选择(elitist expected value selection)、轮盘赌选择(roulette wheel selection)、最优性轮盘赌选择(elitist roulette wheel selection)，并且认为对于结合了决策者模糊目标的多目标整数规划问题来说，最优性期望值选择比较有效。

杂交(crossover) 如果将单点杂交或多点杂交直接应用于双串编码中，一个后代的第 i 个基因的下标 $s(i)$ 数可能与另一个下标 $s(i')(i\neq i')$ 数相同。这种对约束的违背在求解旅行推销员问题或调度问题时也会发生。为了避免这种违背约束的现象，采用了适合于双串编码的改进部分匹配杂交(partially matched crossover,PMX)方法。

双串编码的改进部分匹配杂交过程

第 1 步：选择两个父代 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$，$\boldsymbol{X}$ 的第 i 列为 $(s_X(i),x_{sX(i)})^{\mathrm{T}}$，$\boldsymbol{Y}$ 的第 i 列为 $(s_Y(i),x_{sY(i)})^{\mathrm{T}}$。然后随机确定两个杂交点 $h<k$，设 $i=h$。

第 2 步：根据下面的(2.1)步和(2.2)步通过重新排列 $\boldsymbol{X}$ 的列产生一个新的 $\boldsymbol{X}'$，$\boldsymbol{X}'$ 上面一行在 h 和 k 之间的部分与 $\boldsymbol{Y}$ 中 h 和 k 之间的部分完全相同。

(2.1) 寻找满足 $s_Y(i)=s_X(j)$ 的 j，交换 $\boldsymbol{X}$ 的第 i 列和第 j 列。$i=i+1$。

(2.2) 如果 $i>k$，终止第 2 步。否则返回(2.1)。

然后采用同样的方法重新排列 $\boldsymbol{Y}$ 产生一个新的 $\boldsymbol{Y}'$。

第 3 步：将 $\boldsymbol{X}'$ 中 h 到 k 之间的部分用 $\boldsymbol{Y}$ 中 h 到 k 之间的部分替代，从而产生 $\boldsymbol{X}^*$。采用同样的方法从 $\boldsymbol{Y}'$ 中产生另一个后代 $\boldsymbol{Y}^*$。

该杂交过程用图 4.16 说明。注意染色体中间部分被替换的概率要高于其他部分。这样就引出了双环串的解码过程。该过程中所有的部分具有相同的替换概率。

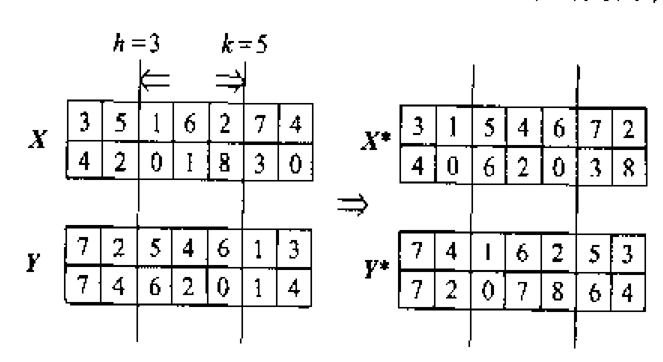


图 4.16 双串编码的 PMX

双环串编码的改进部分匹配杂交过程

第 1 步：选择两个父代 **X** 和 **Y**，**X** 的第 i 列为$(s_X(i),x_{sX(i)})^T$，**Y** 的第 i 列为$(s_Y(i),x_{sY(i)})^T$。

第 2 步：随机确定两个杂交点 h 和 k。如果 $h>k$，$h=h+1$，$k=k-1$。$i=h$。

第 3 步：如果 $i>length$（$length$ 表示串的长度），$n=i-length$。否则 $n=i$。

第 4 步：寻找满足 $s_Y(n)=s_X(j)$的 j，交换 **X** 的第 n 列和第 j 列。$i=i+1$。

第 5 步：如果 $n>k$，将 **X** 中 h 到 k 之间的部分用 **Y** 中 h 到 k 之间的部分替代，从而产生 $\boldsymbol{X}^*$，进入第 6 步。否则返回第 3 步。

第 6 步：对 **Y** 进行相同的操作。产生的 $\boldsymbol{X}^*$ 和 $\boldsymbol{Y}^*$ 就是后代。

图 4.17 举例说明了这个过程。

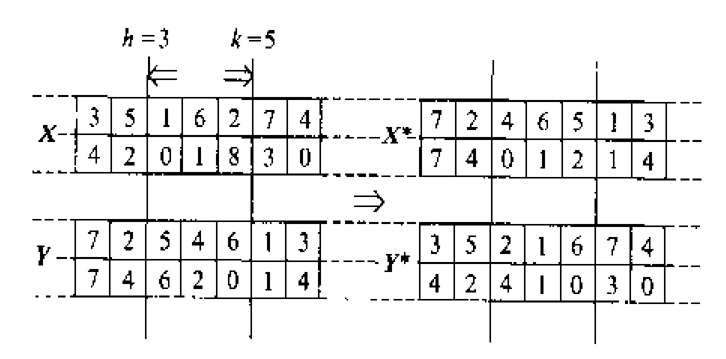


图 4.17 双环串编码的 PMX

变异和反转 在变异操作中，先随机选择变异点，然后随机确定变量 $x_{s(i)}$ 的值。这里 $0\leqslant x_{s(i)}\leqslant v_{s(i)}$（$v_{s(i)}$ 表示 $x_{s(i)}$ 满足所有约束能取的最大值）。进一步还采用了下面过程所定义的反转算子。

变异过程

第 1 步：对双串的每一列，产生一个实随机数 rand(·)∈[0,1]。

第 2 步：如果 $p_m\geqslant$rand(·)，确定该列下部元素的值 $x_{s(i)}$。这里 $0\leqslant x_{s(i)}\leqslant v_{s(i)}$（$v_{s(i)}$ 表示 $x_{s(i)}$ 满足所有约束能取的最大值）。

第 3 步：在种群所有的个体中执行第 1 步和第 2 步。

反转过程

第 1 步：在随机确定两个反转点 h 和 k 后（对于常规双串编码，需要满足 $h<k$），选出从 h 到 k 的子串。

第 2 步：将该子串逆序排列。

第 3 步：将重排列后的子串放入原染色体。

4.5.4 交互式模糊满意方法

这里将一种交互式模糊满意方法结合入遗传算法，从而给决策者根据当前遗传搜索提供的信息重新调整隶属度函数的机会。

第1步：设置初始参考隶属度水平(如果难以确定这些值，则设为1.0)。

第2步：随机产生双串(双环串)编码的 N 个个体的初始种群。

第3步：对每个个体计算适应值，采用基于适应值的复制算子进行复制。

第4步：将每个目标函数的最优解插入当前种群。

第5步：根据杂交概率 p_c 应用杂交算子。

第6步：根据变异概率 p_m 应用变异算子。

第7步：重复第3～第6步指定满足终止条件。此时将具有最大适应值的个体作为最优个体并进入第8步。

第8步：如果决策者对当前最优个体的隶属度函数值和目标函数值满意，停止算法。否则，要求决策者根据当前隶属度函数值和目标函数值更新参考隶属度水平并返回第2步。

在每次交互式过程产生初始种群时，以前遗传交互式过程找到的(近似)最优个体被合并入初始种群。通过采用最优性期望值选择和前面提到的产生初始种群的方法，当前次交互式过程产生的(近似)最优个体很有可能不被前面交互式过程找到的(近似)最优个体所支配。

4.5.5 数值例子

通过下面的过程产生了一个20个变量和15个约束的整数背包问题：

1. 通过均值 $\mu=300$、标准差 $\sigma=50$ 的高斯分布随机数产生每个 a_{ij}。

2. 通过在[1.25,1.75]区间内均匀分布的随机数乘以 $\sum_{j=1}^{n} a_{ij}$ 而产生实数 b_i。

3. 通过与 a_{ij} 相同的方法确定 c_{ij}，其中 c_{1j} 全部设为正数，c_{2j} 全部设为负数。这样就使得 c_{ij} 中正负参半。

上述过程产生的一个问题实例如表4.15所示。

表4.15　带有3个目标和15个约束的整数规划问题实例

a_1	271	331	300	314	256	369	306	226	285	284
	333	379	307	182	312	323	264	288	325	192
a_2	316	202	324	346	328	348	286	263	362	311
	359	340	223	276	376	291	325	296	342	263
a_3	252	265	319	276	158	314	276	277	364	246
	240	316	311	395	371	339	411	300	266	360
a_4	188	288	285	313	306	227	284	273	317	254
	279	340	312	285	286	285	240	399	293	353
a_5	339	252	314	328	269	392	268	367	220	249
	281	295	323	283	286	242	295	270	313	312

续表

a_6	302	267	340	239	377	287	279	282	215	264
	302	421	374	293	227	313	339	353	315	315
a_7	188	260	299	344	352	321	234	317	280	287
	374	353	330	272	297	227	332	291	311	273
a_8	246	349	350	230	297	370	252	265	290	276
	321	237	242	300	260	243	247	411	310	326
a_9	245	299	318	299	307	266	314	301	309	333
	341	315	324	297	344	327	250	367	309	296
a_{10}	211	361	248	217	197	249	300	231	373	320
	256	261	301	302	324	286	391	276	273	298
a_{11}	303	330	328	328	356	284	298	321	269	274
	301	346	289	239	403	252	230	315	192	391
a_{12}	232	277	315	331	262	286	314	369	434	290
	247	318	353	331	315	292	323	250	279	318
a_{13}	198	326	376	289	191	253	239	354	251	298
	315	329	272	255	293	368	320	371	396	293
a_{14}	248	264	275	225	353	310	329	423	306	248
	264	307	251	242	305	274	398	230	204	207
a_{15}	274	335	376	299	334	198	318	305	251	288
	286	273	310	293	226	344	352	236	305	306
c_1	281	327	305	344	371	247	335	336	364	392
	263	280	370	292	231	352	297	254	293	320
c_2	−271	−338	−282	−267	−319	−379	−245	−215	−187	419
	−360	−302	−266	−324	−315	−307	−322	−366	−332	−273
c_3	−315	−293	−245	−291	282	316	226	297	360	182
	−330	394	−345	276	−321	−219	341	−372	250	324
b	8012	7871	8660	8297	9144	10 289	8200	7670	10 091	8522
	9934	10 432	7959	9623	10 108					

仿真结果 对于表 4.15 显示的数值例子，采用常规双串编码和双环串编码各执行 20 次。个体数量为 100，杂交概率 $p_c=0.8$，变异概率 $p_m=0.02$，最大遗传代数 300，每个整数决策变量的最大值是 10。基于 Zimmermann 的方法[705]，使得线性隶属度函数的值变为 0 或 1 的目标函数对(z_1^0,z_1^1)，(z_2^0,z_2^1)，(z_3^0,z_3^1)的值分别为(9200，0)，(0，−9600)，(3000，−9200)。

表 4.16 显示了用两种方法获得的 $\min(\mu_i(z_i(x)))$的最优值、最劣值和平均值，每种方法计算 20 次。从表 4.16 中可以确定，从最劣值和平均值来看，采用双环串编码表示方法的性能比采用双环编码表示方法的性能要好。

表 4.16　20 次计算的结果

编码方式	最　优	最　劣	平　均
双串编码	0.575521	0.563804	0.571542
双环串编码	0.575521	0.567717	0.572829

交互式过程　采用双环串编码表示的遗传算法交互式方法应用于上述问题的仿真结果用表 4.17 表示。采用了与前面数值试验相同的遗传算法参数值和隶属度函数。在本试验中，假定决策者对参考隶属度水平$(\bar{\mu}_1,\bar{\mu}_2,\bar{\mu}_3)$的更新为：(1.0,1.0,1.0)→(0.8,1.0,1.0)→(0.8,1.0,0.9)。表 4.17 所示每次交互产生的(近似)最优解。在第 1 次交互中，获得了参考隶属度水平(1.0,1.0,1.0)时隶属度值为(0.5773,0.5755,0.5917)的(近似)最优解。然而决策者对此解不满意。于是决策者考虑牺牲隶属度函数值 μ_1，即改善隶属度函数值 μ_2 和 μ_3，并由此将参考隶属度值由(1.0,1.0,1.0)改变为(0.8,1.0,1.0)。

表 4.17　交互结果

	μ_1	μ_2	μ_3	$z_1(x)$	$z_2(x)$	$z_3(x)$	解数量
	0.5773	0.5755	0.5917	3898	−5525	−4219	2
第 1 次	0.5763	0.5749	0.5882	3898	−5519	−4177	2
迭代	0.5757	0.5742	0.5848	3907	−5513	−4135	1
$\bar{\mu}_1=1.0$	0.5740	0.5795	0.6522	3919	−5563	−4958	2
$\bar{\mu}_2=1.0$	0.5722	0.6378	0.5736	3923	−5494	−4781	1
$\bar{\mu}_3=1.0$	0.5827	0.5707	0.6331	3839	−5479	−4724	1
	0.5677	0.5762	0.5877	3977	−5532	−4170	1
	0.4917	0.6872	0.7373	4676	−6598	−5939	2
第 2 次	0.4908	0.6867	0.7292	4685	−6598	−5939	1
迭代	0.4945	0.6899	0.6832	4651	−6623	−5335	2
$\bar{\mu}_1=0.8$	0.4829	0.6821	0.7422	4757	6548	−6055	1
$\bar{\mu}_2=1.0$	0.4849	0.6768	0.7169	4727	−6497	−5746	1
$\bar{\mu}_3=1.0$	0.4753	0.6734	0.7484	4827	−6465	−6135	1
	0.4701	0.7227	0.6884	4875	−6938	−5398	1
	0.4824	0.6673	0.7579	4762	−6406	−6246	1
	0.4927	0.6966	0.6310	4667	−6687	−4698	2
第 3 次	0.4933	0.6900	0.6199	4622	−6624	−4563	2
迭代	0.4945	0.6899	0.6832	4651	−6623	−5335	1
$\bar{\mu}_1=0.8$	0.4892	0.6926	0.7369	4669	−6649	−5990	1
$\bar{\mu}_2=1.0$	0.4908	0.6867	0.7293	4685	−6592	−5897	1
$\bar{\mu}_3=0.9$	0.4836	0.6845	0.6108	4751	−6571	−4452	1
	0.4827	0.6828	0.6824	4751	−6571	−4452	1
	0.4799	0.6935	0.7170	4785	−6658	−5747	1

在第 2 次交互中，获得了参考隶属度水平(0.8,1.0,1.0)时隶属度值为(0.4917,0.6872,0.7373)的(近似)最优解。由于决策者希望对于 μ_2 进行进一步改善，他将参考隶属度值由(0.8,1.0,1.0)改变为(0.8,1.0,0.9)。决策者对于第 3 次交互获得的隶属度值(0.4927,0.6966,0.6310)感到满意，于是交互式过程终止。

在这个例子中，更新两次参考隶属度水平后，在第 3 次交互中，得到的结果既是满意解又是 Pareto 最优解。在每次参考隶属度水平下解决增广最小最大问题需要约 69 秒时间，所提出的方法作为交互式方法来说速度很快。不幸的是，在每次交互中，(近似)Pareto 最优解只能在 10 次计算中得到 2 次。考虑到这一点，需要对算法的个体表示、编码算法等方面进行进一步改善。

其他遗传算法在模糊优化中的应用还包括：7.6 节讨论的带有模糊系数的双目标运输问题[412,414,712]，5.4 节讨论的带有模糊目标的多目标可靠性设计[224]，Gen，Tsujimura 和 Li 提出的模糊生产线平衡问题[710]，以及 Sasaki 和 Gen 提出的带有模糊目标和多个选择的多目标背包问题[731]。

第5章　可靠性设计问题

5.1　引言

可靠性优化出现于20世纪40年代晚期并首先应用于通信和运输系统中。大多数早期的工作限于系统某些方面的性能分析。可靠性工程师的目标之一就是增加系统的可靠性。系统的可靠性可以定义为系统在规定条件下指定时间段内正常工作的概率。随着系统越来越复杂，其不可靠行为的后果在费用、努力、寿命等方面变得越加严峻。对于评价系统可靠性的兴趣以及增加产品和系统可靠性的需求变得越来越大。

可靠性问题一般包括可靠性分析、可靠性优化、可靠性提高、可靠性测试、可靠性数据分析、加速测试和生命周期费用[525]。在过去的20年中提出了大量的可靠性设计方法。这些方法可以归类为线性规划(linear programming)、动态规划(dynamic programming)、整数规划(integer programming)、几何规划(geometric programming)、启发式方法(heuristic method)、拉格朗日乘子法(Lagrangian multiplier method)等[521]。文献[382]和[620]给出了关于系统可靠性优化方法的综述。

这些方法的大多数都将描述为非线性整数规划的原始问题转换为0-1线性规划问题。但这种方法会增加原始问题变量和约束的数量。这就要求更多的计算时间和更大的内存空间。因此最好不进行任何转换就求解原始可靠性设计问题。

在Gen和Ida于1993年以及Smith和Coit于1994年[219,229]首先提出用简单遗传算法解决可靠性优化问题以后，许多研究人员研究了基于遗传算法的各种可靠性设计问题[717,719,722,723,725,727]。本章介绍了若干用于求解可靠性设计(reliability design)问题的遗传算法，如网络可靠性设计问题、基于树的网络可靠性设计问题、描述为非线性整数规划(NIP)冗余系统的双目标可靠性设计问题(bicriteria reliability design problems)和带有模糊目标的可靠性设计问题(reliability design problems with fuzzy goals)[229]。这些考虑系统可靠性约束和目标的网络设计问题是许多研究工作的重点，同时也在电信和计算机网络以及电气、煤气、下水道网等领域中有许多应用。

5.2　网络可靠性设计

网络可靠性设计(network reliability design)问题吸引了许多研究人员(如网络设计师、网络分析师和网络管理员)的注意力，其目的是共享昂贵的硬件软件资源并从远端位

置提供对主系统的访问。该问题在诸如电信和计算机网络以及电气、煤气、下水道网等领域中有许多应用。在设计网络系统时,其中很重要的一步就是寻找元件的最佳布局从而优化某些性能判据,诸如费用、传输延迟、生产量或可靠性等。系统的性能判据特别重要,并在很大程度上由网络拓扑决定。在网络设计问题中,考虑可靠性的网络设计是其中典型的问题之一。最优设计问题可以描述为一个组合问题。它通常属于 NP 难问题。与系统设计问题相关的可靠性可以划分为下列两类:

1. *所有终端网络可靠性*(all-terminal network reliability),也称作整体网络可靠性(overall network reliability):网络中的每个节点相互连接的概率,即网络中的每个节点可以与每个其他节点在一段指定的任务时间内通信的概率。

2. *源点-汇点网络可靠性*(source-sink network reliability):源点与汇点连接的可靠性,即网络中的源点可以在一段指定的任务时间内与汇点通信的概率。

考虑这些可靠性的最优网络设计问题求解方法可以分为下面几类:(1)*基于枚举的方法*(enumeration-based approach),(2)*基于启发式的方法*(heuristic-based approach),(3)*基于遗传算法的方法*(genetic algorithm-based approach)。基于枚举的方法仅适用于小规模网络。基于枚举的方法建立在状态枚举、最小路集或最小割集的基础上。粗略地说,这些方法采用了一些枚举组合和缩减技巧。基于枚举的方法枚举出一组关于概率度量相互排斥并且全体穷举的概率事件。Jan,Hwang 和 Chen 提出了一种采用基于分支定界的分解算法来最小化带有最小网络可靠性约束的边费用[325]。Aggarwal 和 Rai 提出了一种基于生成树的方法来评价计算机网络的可靠性[8]。Wilkov 提出了一种用于新可靠性度量方式计算机网络设计的方法。

基于启发式的方法如禁忌搜索、模拟退火等,可以用于大型网络,但不保证最优性。Aggarwal,Chopra 和 Bajwa 在考虑整体可靠性的计算机网络设计问题中应用贪心启发式方法[7]。Chopra 等人在最大化终端可靠性的网络拓扑设计问题中提出了贪心启发式方法[120]。Venetsanopoulos 和 Singh 在满足可靠性约束最小化费用问题中采用了两步启发式方法[638]。Glover,Lee 和 Ryan 提出了一种采用禁忌搜索来进行最小费用网络拓扑设计的方法[246]。Koh 和 Lee 在可靠光纤通信网络设计问题中应用了禁忌搜索方法[372]。Atiqullah 和 Rao 提出了一种应用模拟退火来优化通信网络可靠性的方法[22]。Fetterlof 和 Anandalingam 应用了模糊退火来优化局域网/广域网(LAN-WAN)的网间网设计问题[187]。Pierre 等人提出了一种基于模拟退火的解决计算机通信网络拓扑设计的算法[510]。

最近基于遗传算法的方法作为一种新的考虑可靠性最优网络设计求解方法受到越来越多的重视。Kumer 等人提出了一种用于设计分布式网络拓扑的遗传算法[380]。Kumar,Pathak 和 Gupta 也开发了在设计可扩展计算机网络时考虑直径、距离和可靠性的遗传算法[379]。Walters 和 Smith 针对树状结构网络最优布局设计提出了一种进化算法[647]。Deeter 和 Smith 提出了一种基于遗传算法的方法用于带有可选边(即允许从不

间费用和可靠性的元件中选择边)并且考虑所有终端可靠性(all-terminal reliability)的网络设计问题[153,154]。Dengiz,Altiparmak 和 Smith 尝试将遗传算法应用于考虑所有终端的最优网络设计问题[157,159]。Dengiz,Altiparmak 和 Smith 采用遗传算法求解考虑所有终端网络可靠性的大型骨干通信网络设计问题,并对算法针对所有终端设计问题进行改进以得到效果更好效率更高的优化方法[157,158]

5.2.1　问题描述

一个通信网络可以表示为无向图 $G=(N,E)$,其中 N 个节点和 E 条边分别代表计算机位置和通信电缆。当每对节点 i 和 j 之间至少有1条路径时称图 G 是连通的,连通图的最低要求是有 $(n-1)$ 条边的生成树。n 个节点的图可能存在 $n(n-1)/2$ 条边。下面的符号用于描述所有终端可靠性网络的最优设计问题:n 是节点数量;$x_{ij}\in\{0,1\}$ 是代表节点 i 和 j 之间边的决策变量;$\boldsymbol{x}(=\{x_{12},x_{13},\cdots,x_{n-1,n}\})$ 代表网络设计的拓扑结构;$\boldsymbol{x}^*$ 是迄今为止找到的最优解;p 是所有边的可靠性;q 是所有边的不可靠性(即 $p+q=1$);$R(\boldsymbol{x})$ 是网络设计 $\boldsymbol{x}$ 的所有终端可靠性;$R_{\min}$ 是对网络可靠性的要求;$R_U(\boldsymbol{x})$ 是候选网络的可靠性上界;c_{ij} 是节点 i 和 j 之间边的费用;$c_{\max}$ 是 c_{ij} 的最大值;如果 $R(\boldsymbol{x})<R_{\min}$,则 $\delta=1$,否则 $\delta=0$;E' 是起作用边的集合($E'\subseteq E$);Ω 是所有起作用状态(E')。

假设给定了每个节点的位置,节点不存在可靠性问题,所有的 c_{ij} 和 p 参数都是已知并且是固定的,每条边都是双向的,网络中不存在冗余边,边或是起作用边或是失效边(不起作用边),边的失效在统计上是相互独立的,不存在对边的修复。

网络的最优设计可以描述如下:

$$\min \quad Z(\boldsymbol{x})=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}c_{ij}x_{ij}$$

$$\text{s.t.}\quad R(\boldsymbol{x})\geqslant R_{\min}$$

在任意任务时间内只有 G 中的某些边可能是起作用边。G 的起作用状态就是子图 $G'=(V,E')$。状态 $E'\subseteq E$ 的网络可靠性如下:

$$\sum_{\Omega}\left(\prod_{e\in E'}p_e\right)\left(\prod_{e\in(E\setminus E')}q_e\right)^{①}$$

5.2.2　Dengiz,Altiparmak 和 Smith 的方法

遗传算法已被成功应用于网络可靠性优化问题。本小节将详细介绍 Dengiz,Altiparmak 和 Smith 用于解决网络可靠性优化的方法[153,157]。

① 译者注:该式的含义是对于一种使网络起作用的状态 E',它的概率就是其中所有起作用边的工作概率之积乘以所有不起作用边失效概率之积。对于整个网络来说,将所有这些相互排斥而且全体穷举的状态 E' 的概率加起来就得到网络起作用的概率。

表示 由于每个网络设计 x 可以很容易地转换为二进制串，这个串可以作为遗传算法的染色体，因此遗传算法可以用来求解该问题。染色体中每个元素表示网络设计问题中的一条可能的边。于是每个候选结构中就有 $n\times(n-1)/2$ 个元素组成的字符串。每个元素的值表明特定的边是否与其对应的节点对相连。考虑图 5.1 所示的 5 个节点。注意该问题可能存在 $(5\times4)/2=10$ 条边，但实际只包括 7 条边，其余的 3 条边没有连接。图 5.1 和表 5.1 表示所代表例子的染色体表示如图 5.2 所示。可以看出 x 的下标可用下式表示：

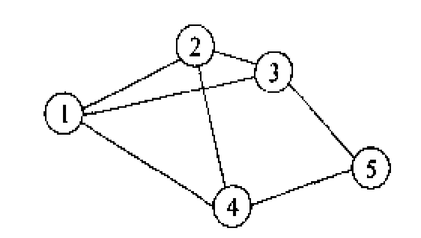


图 5.1 简单网络结构

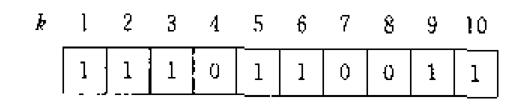


图 5.2 图 5.1 对应的染色体

$$\frac{n(n-1)}{2}-\frac{(n-i)(n-i+1)}{2}+(j-i)$$ [①]

注意，可能的网络结构的解空间数为 $2^{[n\times(n-1)]/2}$。

初始化 初始种群通过带有对高可靠性偏好的随机方法产生，即一系列随机的 2-连接性(*2-connectivity*)。由于 2-连接性通常是较高可靠性网络的特征，因此首先检查2-连接性。2-连接性网络检查通过计算节点的度(即所有节点的度至少为 2，节点的度就是与该节点相连的边的数量)来实现。

对于这样的初始种群来说，选择用来判定边是否存在的概率值对于产生个体的效率来说是非常重要的。表 5.2 表示了所采用的概率区间，这些概率区间由探索性研究来确定。

表 5.1 边的连接性矩阵

$$\begin{bmatrix} - & 1 & 1 & 1 & 0 \\ 1 & - & 1 & 1 & 0 \\ 1 & 1 & - & 0 & 1 \\ 1 & 1 & 0 & - & 1 \\ 0 & 0 & 1 & 1 & - \end{bmatrix}$$

表 5.2 产生初始种群的概率值

n	概率值
10	0.15—0.60
20	0.15—0.50
30	0.10—0.30

因此初始种群可由下列过程产生：

① 译者注：该式的第 1 项表示连接性矩阵上三角阵中所有项的数量，第 2 项表示连接性矩阵上三角阵中第 $n-1$ 行(有 1 个元素)到第 i 行(有 $n-i$ 个元素)共 $(n-i)$ 行中项的数量，第 3 项表示连接性矩阵上三角阵中第 i 行第 $i+1$ 列到第 j 列中项的数量。也就是说，用所有上三角阵中项的数量减去第 i 行以后的项的数量再加上第 i 行项的数量就得到当前基因的下标。

第1步：根据表5.2的边概率随机产生初始种群。

第2步：如果第1步产生的染色体不满足2-连接性要求，抛弃之，重复第1步。

第3步：如果已经产生了 *pop_size* 个染色体(其中 *pop_size* 是种群规模)，终止过程。

适应值　惩罚函数的目的是将优化算法引导到临近的最优可行解上。这里采用的目标函数是将网络中所有边的费用相加，并附上网络在不满足最低可靠性要求时的二次惩罚函数。考虑了不可靠性的目标函数定义如下：

$$Z(x) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}x_{ij} + \delta[C_{\max}(R(x) - R_{\min})]^2 \tag{5.1}$$

适应值函数 eval(x)采用下面的方程定义：

$$\text{eval}(x) = Z_{\max} - Z(x) \tag{5.2}$$

其中 $Z_{\max}$是当前种群中最大(最差)的目标函数值。

选择　选择过程采用旋转轮盘 *pop_size* 次的方法，每次选择一个染色体成为后代。

杂交　采用了单点杂交操作。这种方法如图5.3所示。

变异　采用了位翻转变异操作，该方法按位进行。假设变异点指向下标2的元素。通过位翻转产生的后代如图5.4所示。

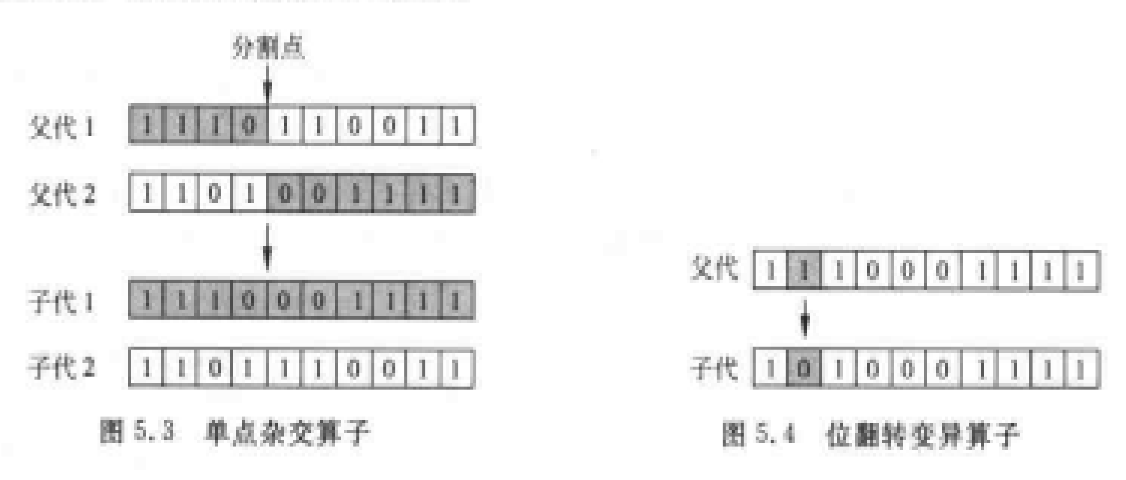


图5.3　单点杂交算子　　　　图5.4　位翻转变异算子

可靠性估计　关于网络设计问题另一个活跃的研究领域就是计算或估计网络的可靠性。有两种主要的方法：(1)通过分析方法来准确计算(exact calculation)；(2)通过改进的 Monte Carlo 模拟(Monte Carlo simulation)来估计。

已知的所有终端网络可靠性计算的分析方法在最差情况下的计算时间会随着网络规模的增大而指数增加。因此这些分析方法对于大型问题来说并不被推荐。Ball 和 Van Slyke 提出了一种通过回溯来枚举改进割集的方法，可以用来计算一个无向图的整体可靠性[41]。Hanzhong 和 Dongkui 提出了一种新的用于计算复杂网络可靠性的方法[281]。Mandaltsis 和 Kontolen 给出了一种计算计算机网络整体可靠性的方法，该方法具有分级路径策略[435]。Liu 等人提出了用于计算网络整体可靠性的方法[420]。

对于所有终端网络可靠性问题，如果网络达到完全连接状态，模拟方法的效率一般会

大大降低，从而实现很困难。然而模拟方法对于大型网络来说还是合适的，原因在于随网络规模的增加，这种方法所需的计算时间仅比线性关系多一点。Cancela 和 Khadiri 在 Monte Carlo 方法中采用了递归偏差削减算法来估计通信网络可靠性[83]。Fishman 研究了 Monte Carlo 采样计划，将其用于估计网络可靠性。他还分析了 4 种 Monte Carlo 方法来估计源点-汇点连通性的可靠性[190,191]。Kamat 和 Riley 检验了一种带有基于事件 Monte Carlo 模拟的可靠性判断方法[343]。Kumamoto，Tanaka 和 Inoue 采用 Monte Carlo 方法提出了一种对系统可靠性的有效评价算法[378]。Yeh，Lin 和 Yeh 针对网络可靠性估计提出了一种新的 Monte Carlo 方法[662]。

由于遗传算法是一种迭代算法，每代中需要对多个候选网络进行评价，因此传统的精确计算方法不适合与遗传算法一起工作。因此采用了下述的候选网络设计过程：

第 1 步：连接性检查。针对所有新的网络设计采用深度优先和广度优先搜索进行生成树的连接性检查。如果一个图任意两个节点都是连通的，则该图就是连通的。

第 2 步：2-连接性检查。对于通过第 1 步的网络设计，执行 2-连接性检查。通过验证是否所有节点的度都至少是 2 来完成 2-连接性检查。

第 3 步：计算上界。对于通过前述步骤的网络，采用 Jan 上界来计算 $R_U(\boldsymbol{x})$[324,325]。

该上界用于计算除了 $\boldsymbol{x}^*$ 之外所有网络的目标函数。满足 $R_U(\boldsymbol{x}) \geqslant R_{\min}$ 并且迄今为止具有最小费用的网络送入 Monte Carlo 模拟，目的是利用有效的 Monte Carlo 方法来更为精确地估计网络可靠性。对于每个候选网络 Monte Carlo 模拟过程执行 3000 次迭代。

Jan 上界 设图 G 有 n 个节点，度次序为 d。Jan 上界的计算如下：

$$R(G) \leqslant H(d)$$

$$H(d) \equiv 1 - \left[\sum_{i=1}^{n} q^{d_i} \prod_{k=1}^{m_i} (1 - q^{d_k - 1}) \prod_{k=m_i+1}^{i-1} (1 - q^{d_k}) \right]^{①}$$

$$m_i = \min\{d_i, i-1\}$$

比如设 $d = \{2,2,2,2\}, q = 0.1$ 则

$$R(G) \leqslant 1 - (0.1^2 + 0.1^2 \times 0.9 + 0.1^2 \times 0.9 \times 0.9 + 0.1^2 \times 0.9 \times 0.9 \times 0.99)$$
$$= 0.9649^{②}$$

① 译者注：$H(d)$ 表示的是网络可靠性的上界，该式方括号中表示的就是网络不可靠性的下界(即系统失效的最小概率)。求和号中每一项都是使系统失效相互排斥而且全体穷举的事件。

② 译者注：不等式右端的括号中，第一项是与第 1 个节点相邻所有边均失效的概率。第 2 项是与第 2 个节点相邻所有边均失效同时与第 1 个节点相邻 1 条边工作的概率。第 3 项是与第 3 个节点相邻所有边均失效同时第 1 个和第 2 个节点均有 1 条相邻边工作的概率。第 4 项是与第 4 个节点相邻所有边均失效同时第 1，2，3 个节点中有 2 个节点有 1 条相邻边工作，另外 1 个节点所有边均工作的概率。可以看出，这些实际是相互排斥并且全体穷举的。它们的概率之和构成了系统失效的最小概率。

整体算法

第1步：设置参数。设置种群规模(pop_size)、变异率(p_m)、杂交率(p_c)和最大代数(max_gen)，并设初始遗传代数 $gen=0$。

第2步：初始化。

(2.1) 根据表5.2的边概率随机产生初始种群。如果产生的染色体不满足2-连接性要求，抛弃并重新产生。

(2.2) 采用式(5.1)、式(5.2)和Jan上界 $R(\boldsymbol{x})$ 计算种群中每个候选网络的适应值(除满足 $R_U(\boldsymbol{x})\geqslant R_{min}$ 的最小费用网络之外)。

(2.3) 对于最小费用网络 x^*，采用Monte Carlo模拟来仿真出 $R(\boldsymbol{x})$。

第3步：通过轮盘赌选择机制从当前种群中选出两个候选网络。

第4步：执行杂交和变异。为了获得两个子代候选网络，对选出的两个染色体进行参数为 p_c 的杂交操作和参数为 p_m 的变异操作。

第5步：检验每个新子代的2-连接性。抛弃任何不满足2-连接性的个体。

第6步：计算Jan上界。对每个个体计算 $R_U(\boldsymbol{x})$ 并得到其目标函数值。

第7步：检查子代个体数量。如果 $n<pop_size-1$，返回第3步；否则进入第8步(其中 n 代表新产生个体的数量)。

第8步：建立新种群。在保留上一代的 $\boldsymbol{x}^*$ 的基础上将新一代替换掉老一代。

第9步：评价

(9.1) 根据 $Z(\boldsymbol{x})$ 的增序对新一代个体进行排序。

(9.2) 如果 $Z(\boldsymbol{x}_p)<Z(\boldsymbol{x}^*)$，$p=1,2,\cdots,pop_size$，采用Monte Carlo模拟来估计该网络的系统可靠性，并设 $\boldsymbol{x}^*=\boldsymbol{x}_p$，否则进入(9.3)步。

(9.3) 计算新一代中每个网络的 $\mathrm{eval}(\boldsymbol{x})$。

第10步：执行终止验证。如果 $gen<max_gen$，$gen=gen+1$，返回第3步进入下一代。如果 $gen=max_gen$，终止算法。

数值例子 Dengiz，Altiparmak和Smith采用的测试例子总结在表5.3中[156,157]。这些问题包括完全连接网络和非完全连接网络($\boldsymbol{E}$ 中仅有一个子集可以进行选择)。网络的节点数从5到20。非完全连接网络的可能边随机产生，数量为 $1.5N$。所有网络的边费用均在[1,100]中随机产生。用遗传算法优化每个问题10次，每次采用不同的随机数种子。Jan，Hwang和Chen通过分支定界(B&B)方法得到的最优解如表5.3所示[325]。

表5.4列出了每个问题的搜索空间和遗传算法进行实际搜索的比例(每次运行遗传算法进行了 $pop_size*max_gen$ 次搜索)。随着问题规模的不同，max_gen 从30到20 000不等。这个比例是遗传算法搜索的上界，原因是遗传算法可以(经常)访问到以前进化搜索访问过的点。对于大型问题，遗传算法仅搜索了可行解很小的一部分，就已经得到最优或近似最优解。表5.4还比较了用Monte Carlo估计网络可靠性的有效性。通过

Jan,Hwang 和 Chen 给出的回溯算法[325]得到了准确的网络可靠性,将其与遗传算法找到最优解所对应的网络可靠性相比较。Monte Carlo 方法的可靠性估计(reliability estimation)是无偏的,总是在准确网络可靠性 1%的范围之内。由于 Monte Carlo 方法的计算时间不随网络规模变化,同时估计的准确性也不随 E 的增加而降低,这种仿真估计可以成为准确网络可靠性估计很有效的代用品。

表 5.3 B&B 和遗传算法的结果比较

问题	N	E	p	R_{min}	B&B 最优费用	遗传算法		
						最优费用	平均费用	偏差系数
1	5	10	0.90	0.95	201	201	201.0	0
2	8	28	0.95	0.95	179	179	180.3	0.0228
3	10	45	0.90	0.95	197	205	206.6	0.0095
4	20	190	0.95	0.95	—	926	956.0	0.0304
5	14	21	0.90	0.90	1063	1063	1076.1	0.0129
6	16	24	0.90	0.95	1022	1022	1032.0	0.0204
7	20	30	0.95	0.90	596	596	598.6	0.0052

注:问题 1~4 是完全连接网络,问题 5~7 是非完全连接网络。

表 5.4 搜索规模和可靠性估计的比较

问题	搜索空间	搜索过的空间	搜索所占的比例	R_{min}	$R(x)$	$\hat{R}(x)$	区别的百分比
1	1.02×10^{6}	6.00×10^{2}	5.86×10^{-4}	0.95	0.9579	0.9604	0.261
2	2.68×10^{8}	2.00×10^{4}	7.46×10^{-5}	0.95	0.9637	0.9645	0.083
3	3.52×10^{13}	8.00×10^{4}	2.27×10^{-9}	0.95	0.9516	a	
4	1.57×10^{27}	2.00×10^{5}	1.27×10^{-22}	0.95	b	0.9925	
5	2.10×10^{6}	1.50×10^{4}	7.14×10^{-4}	0.90	0.9035	0.9035	0.000
6	1.68×10^{7}	2.00×10^{4}	1.19×10^{-4}	0.95	0.9538	0.9550	0.126
7	1.07×10^{9}	3.00×10^{4}	2.80×10^{-5}	0.90	0.9032	0.9027	−0.055

a:遗传算法未发现最优解。

b:网络过于庞大,以致无法准确计算可靠性。

5.2.3 Deeter 和 Smith 的方法

Deeter 和 Smith 提出了考虑所有终端可靠性(all-terminal reliability)时设计网络的遗传算法[153,154]。这类问题通常假设网络中所有边的可靠性相同,其费用取决于边所连接的两个节点。对于每对节点来说,仅有一种可靠性和费用的选择。Deeter 和 Smith 提出的方法大大扩展了以前的研究工作,这种方法允许从带有不同费用和可靠性的不同元件中选择边。

定义下面的符号来表示允许从不同边选项中进行选择的网络最优设计：k 是边连接可选的数量，t 是节点间的选项，$x_{ij}(x_{ij}\in\{0,1,2,\cdots,k-1\})$是节点 i 和 j 之间的边选项，$p(x_{ij})$是边选项的可靠性，$c(x_{ij})$是边选项的费用。

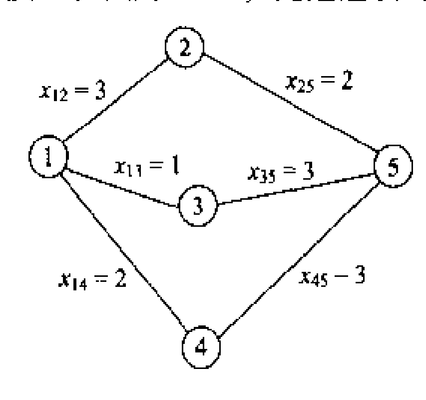


图 5.5 网络结构例子

表示 由于每个网络设计可以轻松地表示为整数向量，这个向量可以作为遗传算法的染色体。染色体的每个元素表示网络设计问题中一条可能的边。于是每个候选结构 $\boldsymbol{x}$ 中存在 $n\times(n-1)/2$ 个向量元素。每个元素的值表示连接两节点之间的特定边采用哪种类型的连接。图 5.5 和表 5.5 表示考虑有 5 个节点和 $k=4$ 个连接选项的问题。注意这个例子有$(5\times4)/2=10$条可能的边，但实际上只有 6 条；其余 4 条的连接选项 $k=0$。在表 5.5 表示的边矩阵中，可以看出矩阵是对称的。因此用来记录这个特殊结构的所有信息都在矩阵的上三角部分。采用连接矩阵每行的方式将这种信息记录在长度为$(5\times4)/2=10$ 的染色体(向量)中，如图 5.6 所示。注意在染色体每个位置上可能的取值为 $0,1,\cdots,k-1$。可能的网络结构解空间为 $k^{(n\times(n-1))/2}$。

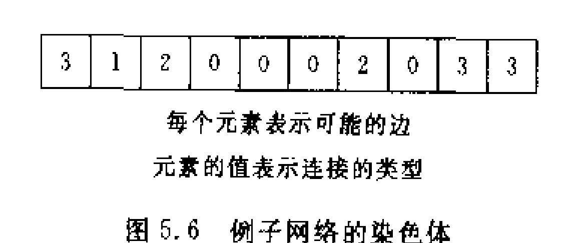


图 5.6 例子网络的染色体

表 5.5 边的连接性矩阵

$$\begin{bmatrix} - & 3 & 1 & 2 & 0 \\ 3 & - & 0 & 0 & 2 \\ 1 & 0 & - & 0 & 3 \\ 2 & 0 & 0 & - & 3 \\ 0 & 2 & 3 & 3 & - \end{bmatrix}$$

适应值 遗传算法试图在满足或超过事先指定网络可靠性 $R_{\min}$ 的基础上寻找最小费用的网络结构。适应值函数的构造需要考虑不可行网络的结构。不可行解可能包含有用的部分。因此将两个不可行解或一个不可行解和一个可行解进行繁殖可能得到好的可行解。同时由于最小费用网络仅有一个约束需要满足或近似满足，最优解存在于可行与不可行设计的边界上。适应值函数如下：

$$Z_p(\boldsymbol{x}) = Z(\boldsymbol{x}) + Z(\boldsymbol{x}^*)[1 + R_{\min} - R(\boldsymbol{x})]^{r_p \cdot (pop_size \cdot gen)/50} \text{①}$$

其中 $Z_p(\boldsymbol{x})$是惩罚后的费用，$Z(\boldsymbol{x})$是未惩罚的费用，$Z(\boldsymbol{x}^*)$是种群中最优可行解的费用，r_p 是惩罚率，pop_size 是种群规模，gen 是遗传代数。

① 译者注：如果 $R(x)\geqslant R_{\min}$(即满足可靠性约束)，惩罚项是很小的数；如果 $R(x)\leqslant R_{\min}$(即不满足可靠性约束)，惩罚项是很大的数。$Z(x^*)$ 的含义是如果当前种群中最优解比较好，则对于不可行解的惩罚较小。gen 项的含义是随着进化过程对不可行解的惩罚逐渐增加。

初始种群 种群的初始化由下面的过程随机产生：

第 1 步：根据边的可靠性和费用选项随机产生种群。

第 2 步：将初始种群送入可靠性计算过程。

第 3 步：将初始种群送入费用计算过程。如果存在不可行染色体，该染色体受到惩罚。

第 4 步：寻找最优初始染色体。如果没有染色体是可行的，则记录最优的不可行染色体；否则记录最优可行染色体。

选择 采用基于排序的二次过程来选出两个父代进行繁殖。

第 1 步：根据染色体适应值的水平进行降序排列，即 1 表示适应值最高的水平(费用最低)。

第 2 步：产生 0 和 $\sqrt{pop_size}$ 之间的随机数。

第 3 步：对该随机数平方。

第 4 步：截去平方后的数。

第 5 步：对于截去的数加 1。

第 6 步：将当前随机数对应的染色体选出。然后返回第 2 步选择另一个父代。如果两次选出的是同一父代，则重复该过程直到产生不同的父代为止。[①]

比如，设种群规模为 20。选择过程如下：

第 1 步：根据 20 个个体的适应值水平进行降序排列。

第 2 步：产生 0 和 $\sqrt{20}$ 之间的随机数。

第 3 步：假设随机数是 2.5，对其平方得到 6.25。

第 4 步：6.25 进行截断得到 6。

第 5 步：6+1 得到 7。

第 6 步：从种群中选中第 7 个染色体。

杂交 采用了均匀杂交(uniform crossover)。这种杂交从两个父代解中随机提取元素来形成子代对应的元素，如图 5.7 所示。

第 1 步：通过选择操作选出两条染色体。

第 2 步：从一条染色体中随机选择一个元素形成子代中对应的元素。

第 3 步：重复第 2 步直到子代所有元素都被赋值。

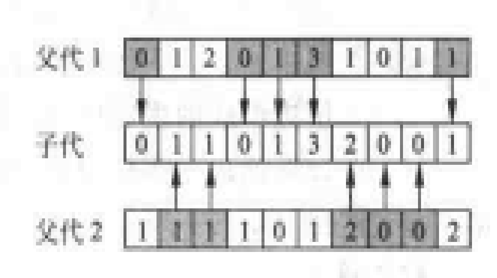


图 5.7 均匀杂交算子

变异 变异是向种群中添加新遗传材料的一种方式。这种操作可以使遗传算法避免陷入局部最优中。根据种群中变异的百分比 p_{m1} 来决定一个个体是否进行变异。一旦一个个体需要进行变异，其向量中每个元素的变异概率由 p_{m2} 决定。如果

① 译者注：从该方法的实施过程可以看出，适应值较低的个体被排列在前面，因此具有较高的被选择概率。

$p_{m2}=0.3$,选出个体的每个元素均有0.3的概率进行变异。

第1步：产生实随机数 $r_1, r_1\in[0,1]$。

第2步：如果 $r_1<p_{m1}$,染色体进行变异,进入第3步；否则进入第6步。

第3步：产生随机数 $r_2, r_2\in[0,1]$。

第4步：如果 $r_2<p_{m2}$,进入第5步,否则进入第6步。

第5步：如果 $x_{ij}=0$,从$\{1,2,\cdots,k-1\}$中随机选择一个数作为新的元素值,如果 $x_{ij}=t$ $(\neq 0)$,则以概率0.5从$\{1,2,\cdots,k-1\}-\{t\}$中随机选出一个数或以概率0.5选择0作为新的元素值。

第6步：结束变异操作。

一次变异操作可以用图5.8表示。

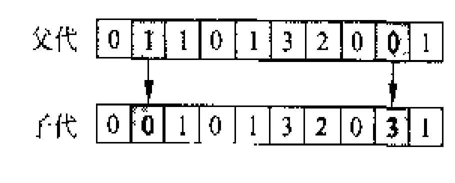


图5.8　变异算子

可靠性计算　鉴于问题处于可以计算的规模内,故采用了回溯算法计算系统准确的不可靠性 $1-R(x)$。采用了下面的回溯算法,一个堆栈中所有边的概率就是所有不起作用边失效概率之积乘以所有起作用边正常工作概率之积。

第1步：初始化。将所有边设为自由边,创建堆栈 S 并初始化为空集。

第2步：产生改进的割集。

(2.1) 寻找一组自由边,使其能够与堆栈中的不起作用边联合构成网络的割集。

(2.2) 将(2.1)步找到的所有自由边标记为不起作用边,并将其加入堆栈。

(2.3) 现在堆栈表示一个改进的割集,计算这个割集的概率并将其加入整个系统的不可靠性中。

第3步：回溯。

(3.1) 如果堆栈空,进入第4步。否则进入(3.2)步。

(3.2) 从堆栈顶端取走一条边。

(3.3) 如果该边为不起作用边,同时如果将其标记为起作用边后网络存在由起作用边组成的生成树,则将该边标记为自由边,返回(3.1)步。

(3.4) 如果该边为不起作用边,同时如果将其标记为起作用边后网络不存在由起作用边组成的生成树,将其标记为起作用边并重新放入堆栈中,返回第2步。

(3.5) 如果该边为起作用边,将其标记为自由边并返回(3.1)步

第4步：返回网络的不可靠性并终止过程。

整体算法

第1步：设置参数。设置种群规模(pop_size)、种群中个体的变异率(p_{m1})、个体中元素的变异率(p_{m2})、惩罚(r_p)和最大代数(max_gen),并设初始遗传代数 $gen=0$。

第2步：初始化。

(2.1) 随机产生初始种群。

(2.2) 将初始种群送入可靠性计算过程。

(2.3) 将初始种群送入费用计算过程。如果存在不可行染色体,该染色体受到惩罚。

(2.4) 寻找最优初始染色体。如果没有染色体是可行的,则记录最优的不可行染色体。

第 3 步:进行选择。

(3.1) 将最优个体插入到新种群中。

(3.2) 采用基于排序的二次选择方法从当前种群中选择两个不同的候选染色体。

第 4 步:执行杂交和变异。为获得 2 个子代,应用均匀杂交算子。产生一个子代后对其进行变异。

第 5 步:检查子代数量。如果 $n<pop_size-1$,返回第 3 步;否则进入第 6 步。其中 n 表示新子代的数量

第 6 步:建立新种群。用产生的新子代替代其父代。

第 7 步:评价

(7.1) 将新种群送入可靠性计算过程。

(7.2) 计算新种群中每个染色体的 eval(x)。如果存在不可行染色体,对其进行惩罚。

第 8 步:搜索最优染色体。保存最优染色体。如果没有染色体是可行的,则保存最优的不可行染色体。

第 9 步:执行终止验证。如果 $gen<max_gen$, $gen=gen+1$,返回第 3 步进入下一代。如果 $gen=max_gen$,终止算法。

数值例子 Deeter 和 Smith 采用了 5 节点多水平的问题[153,154]。该问题基于 Jan, Hwang 和 Chen 的测试问题[325](用表 5.6 表示)。注意每单元的费用随着可靠性的增加而增加的程度超过线性。该问题的搜索空间规模为 1 048 576,考虑了 7 种不同的系统可靠性水平。由于这是一个较小的问题,可以针对 7 种可靠性水平枚举所有的解来获得最优解。这些结果用表 5.7 表示。注意每个最优解都采用了两个或更多的边连接水平。这表明通过允许不同的边可靠性可以改善设计。

表 5.6 边单元的可靠性和费用

连线类型	可靠性	费用/单位距离
0	0.00	0
1	0.85	4
2	0.90	16
3	0.95	48

表5.7　测试问题的最优解

R_{min}	费用	$R(x)$	染色体
0.99900	5522	0.99908	3323333323
0.99500	4352	0.99518	3113311313
0.99000	3754	0.99052	3203322303
0.95000	2634	0.95353	3003302303
0.93125	2416	0.93361	2003301303
0.90000	2184	0.91854	3003300303
0.85000	1904	0.85536	3003200203

在探索性试验运行以后，遗传算法的参数设置如下：$pop_size=40$，$p_{m1}=0.25$，$p_{m2}=0.25$，$max_gen=6000$，$r_p=6$。对于每个系统可靠性采用不同随机选择的随机数种子进行10次计算。由于遗传算法是随机搜索方法，这种方法验证了算法的有效性。

5.3　基于树的网络可靠性和局域网设计

许多人研究过通信网络的拓扑设计问题。在许多实际应用问题中，如何有效地设计满足特定约束的网络是非常重要的问题。特别是在计算机网络系统中，局域网(LAN)是满足局部环境中用户需求的常见通信基础结构。这些计算机网络通常包含若干通过网桥相互连接的LAN段。使用这些透明的网桥需要在LAN段之间存在无环(loop-free)路径[174]。因此采用生成树拓扑就成为现行的网络结构。在设计计算机网络系统拓扑的同时，还需要考虑元件的最优布局以最优化某些性能判据(如费用、消息延迟、通信量或可靠性)。网络拓扑设计问题包括两个主要方面：聚类和路径。聚类问题包括LAN中需要划分多少段(类)以及如何根据LAN段(类)来分配用户(工作站)。路径问题就是如何确定段的相互连接，使其构成生成树拓扑。

由于遗传算法被视作网络优化问题潜在的优化方法，受到了广泛的关注，经常用来解决实际问题[219,455]。选择最优LAN拓扑是非常复杂的组合优化问题，属于NP难问题。因此基于遗传算法的启发式算法受到关注。采用遗传算法来解决网络拓扑设计问题已经有若干篇文章。Elbaum和Sidi采用基于Huffman树的遗传算法解决LAN的拓扑设计[174]。Gen，Ida和Kim提出了用于解决考虑连接费用和平均消息延迟的双目标网络设计问题(bicriteria network design problem)的遗传算法[223,225]。

5.3.1　双目标网络拓扑设计

考虑连接了m个用户(工作站)的LAN。比如图5.9表示了有5个服务中心和18个用户的LAN。我们还假设$n \times n$维服务中心拓扑矩阵(service center topology matrix)$\boldsymbol{X}_1$

(用于表示服务中心之间的连接)。该矩阵的元素 x_{1ij} 表示为

$$x_{1ij}=\begin{cases}1, & \text{如果中心 } i \text{ 和 } j \text{ 相连}\\0, & \text{否则}\end{cases}$$

假设 LAN 被划分为 n 段(服务中心或类)。用户被分配到这 n 个服务中心中。$n\times m$ 维聚类矩阵(clustering matrix)$\boldsymbol{X}_2$ 表示了用户属于那个中心,即

$$x_{2ij}=\begin{cases}1, & \text{如果用户 } j \text{ 属于中心 } i\\0, & \text{否则}\end{cases}$$

一个用户只能属于一个中心,因此 $\forall j=1,2,\cdots,m,\sum_{i=1}^{n}x_{2ij}=1$。我们定义 $n\times(n+m)$ 维生成树矩阵(spanning tree matric)$\boldsymbol{X}([\boldsymbol{X}_1\ \boldsymbol{X}_2])$。

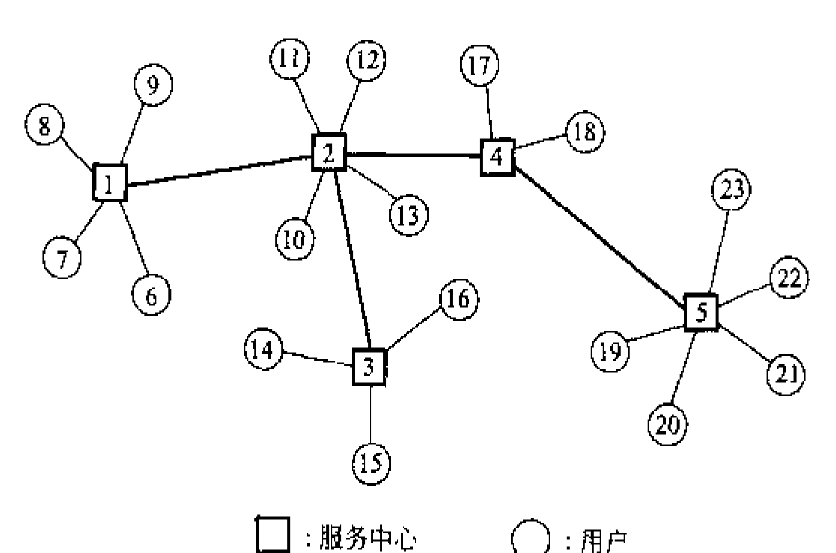


图 5.9 简单 LAN 结构

服务中心通过网桥相互连接起来。从源中心 i 到目标中心 j 之间的通信通过网桥传送。中心 i 和 j 之间的路径可能仅包括一个连接两个中心的网桥,或当中心之间没有直接连接时的多个网桥。后一种情况中,通信通过连接路径上的网桥和中心进行。如前所述,仅考虑采用透明网桥。选择透明网桥就要求网络配置为生成树拓扑结构。

一个中心就是带有已知容量的 LAN 段。一个 LAN 段可以是令牌环网、以太网或其它类型的结构。显然每种类型的 LAN 段的行为和性能互不相同。然而所有这些结构的平均延迟随着负荷的增长定性地以相同方式增加。随着段上负荷的增加,平均延迟增加。当负荷接近段容量时,延迟接近无穷。

双目标 LAN 拓扑设计问题可以描述为下面的非线性 0-1 规划模型:

$$\max \quad R(\boldsymbol{X}) \tag{5.3}$$ [①]

$$\min \quad \sum_{i=1}^{n-1}\sum_{j=i-1}^{n} w_{1ij}x_{1ij} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{m} w_{2ij}x_{2ij} \tag{5.4}$$ [②]

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{2ij} \leqslant g_i, \quad i = 1,2,\cdots,n \tag{5.5}$$ [③]

$$\sum_{i=1}^{n} x_{2ij} = 1, \quad j = 1,2,\cdots,m \tag{5.6}$$ [④]

其中 $R(\boldsymbol{X})$是网络可靠性，w_{1ij}是中心 i 和 j 之间连接的权重，w_{2ij}是中心 i 和用户 j 之间连接的权重，g_i 是连接到中心 i 的最大容量数。

表示和初始化　遗传表示是用来表示问题候选解的一种数据结构。通常不同的问题采用不同数据结构和遗传表示。两种类型的数据结构可以用来表示现行的 LAN 结构：(1)服务中心和用户都用 Prüfer 数来表示[223,225]；(2)服务中心用 Prüfer 数来表示，用户用描述用户在服务中心中的聚类串来表示。

图枚举中的一个经典定理就是 Cayley 定理。该定理表明 k 个节点的完全图有 $k^{(k-2)}$ 棵不同的树。Prüfer 通过在树和所有由 $k-2$ 个数字组成的字符串之间建立一一对应关系为 Cayley 定理给出了一个构造性的证明[699]。这意味着只需要 $k-2$ 个数字排列就可以惟一地表示一棵树，其中每个数字是 1 到 k 之间的整数。这个排列通常被称作 Prüfer 数。对于任意的树来说至少存在 2 个叶节点[576]。基于这种情况，可以很方便地构造下面的编码。

Prüfer 数的编码过程

第 1 步：设 i 是树 T 中具有最小下标的叶节点。

第 2 步：设惟一与 i 节点相连的 j 节点成为编码中的第一个数字。编码数字的顺序从左到右。

第 3 步：将 i 节点和 i 和 j 之间的边从图中移去，这样就构成了有 $k-1$ 个节点的树。

第 4 步：重复上述过程直到仅乘一条边。这样就产生了一个 Prüfer 数或一个 $k-2$ 个数字的编码(这些数字的取值为从 1 到 k)。

也可以根据下面的过程。从 Prüfer 数产生惟一的树(图 5.10)。

1	1	2	2

Prüfer 数

图 5.10　树及其 Prüfer 数

Prüfer 数的解码过程

第 1 步：设 $\bar{P}$ 为原始 Prüfer 数，$\bar{P}$ 是所有未包含在 P 中

---

① 译者注：表示最大化网络可靠性。

② 译者注：表示最小化中心之间相互连接的权重(费用)和中心与用户之间相互连接的权重(费用)。

③ 译者注：表示与每个中心相连的用户数量的限制。

④ 译者注：表示每个用户只连接到一个中心上。

的节点的集合，$\bar{P}$ 用来在构造树的时候表示合格的节点。

第 2 步：设 i 是最小下标的合格节点。设 j 是 P 中最左边的数字。在树中添加从 i 到 j 的边。将 i 从 $\bar{P}$ 中移去，j 从 P 中移去。如果 j 在 P 的剩余部分中都不再出现，将其放入 $\bar{P}$。重复这一步直到 P 中不包含数字为止。

第 3 步：如果 P 中不包含数字，则 $\bar{P}$ 中正好有两个合格的节点 r 和 s。在树中 r 和 s 之间添加一条边以构成一棵 $k-1$ 条边的树。

这种表示有两种基因型：服务中心用 Prüfer 数表示，用户用聚类串表示。染色体在初始化过程中随机产生(图 5.11)。服务中心包括在[1,n]区间中随机产生的 $n-2$ 个数字(Prüfer 数)，用户包括在[1,n]区间中随机产生的 m 个数字(聚类串)，该串表明如何将用户分配给服务中心以确保每个用户属于一个特定的中心。

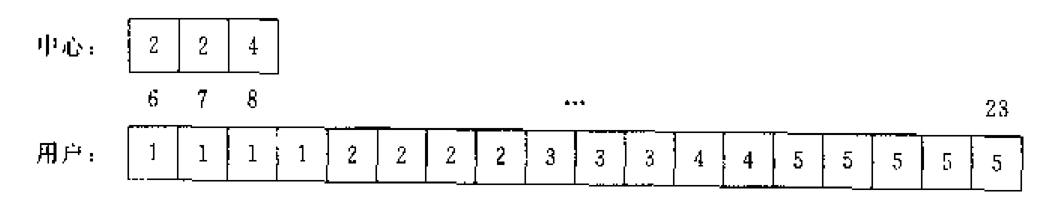


图 5.11 图 5.9 的表示

评价 单目标 LAN 拓扑问题可以很轻松地通过根据目标函数来计算每个染色体的适应值。但在双目标 LAN 拓扑设计问题中，我们必须处理 Pareto 解。这里采用了加权和方法来构造适应值函数，该方法将两个目标函数组合成一个完整的适应值函数。

第 1 步：将用 Prüfer 数和聚类串表示的染色体转换为生成树矩阵 $\mathbf{X}$。

第 2 步：计算目标函数值[$f_i(\mathbf{X})$, $i=1,2$]。

第 3 步：根据每个目标函数值选出包含最小 $f_i^{\min}$(或最大 $f_i^{\max}$)的解点，并将其与以前保存的解点进行比较，将最好的解点加以保存。

$$f_i^{\min(t)} = \min_k\{(f_i^{\min(t-1)}, f_i^{(t)}(\mathbf{X}_k)) \mid k = 1,2,\cdots,chr_size\}, \quad i = 1,2$$

$$f_i^{\max(t)} = \max_k\{(f_i^{\max(t-1)}, f_i^{(t)}(\mathbf{X}_k)) \mid k = 1,2,\cdots,chr_size\}, \quad i = 1,2$$

其中 $f_i^{\min(t)}$ 和 $f_i^{\max(t)}$ 分别是第 i 个目标在第 t 代的最小目标函数值和最大目标函数值。$f_i^{(t)}$ 是第 k 个染色体第 t 代时第 i 个目标函数值。chr_size 是 pop_size 加上遗传操作产生的后代数的数值。

第 4 步：计算权重系数如下：

$$\beta_i = \frac{\alpha_i}{\sum_{i=1}^{2}\alpha_i}, \quad i = 1,2$$

其中

$$a_i=\frac{f_i^{\max(t)}-f_i^{\min(t)}}{f_i^{\max(t)}},\quad i=1,2$$ [①]

第5步：对每个染色体计算适应值如下：

$$\mathrm{eval}(\boldsymbol{X}_k)=\sum_{i=1}^{2}\beta_i d_i(\boldsymbol{X}_k),\quad k=1,2,\cdots,chr_size$$

其中 $d_i(\boldsymbol{X}_k)$是第 i 个目标函数的归一化后的值，表示如下：

$$d_i(\boldsymbol{X}_k)=\begin{cases}\dfrac{f_i^{(t)}(\boldsymbol{X}_k)-f_i^{\min(t)}+\gamma}{f_i^{\max(t)}(\boldsymbol{X}_k)-f_i^{\min(t)}+\gamma}, & \text{对最大化问题}^{②}\\[2ex] \dfrac{f_i^{\max(t)}(\boldsymbol{X}_k)-f_i^{(t)}+\gamma}{f_i^{\max(t)}(\boldsymbol{X}_k)-f_i^{\min(t)}+\gamma}, & \text{对最小化问题}\end{cases}$$

γ 是小的正实数，通常限制在开区间(0,1)中，用来防止产生被零除错误，又可以使得基于适应值比例的选择过程有可能调整成为纯粹随机选择。[③]

如果读者需要参考这种归一化方法和权重和评价的更多信息，请参考文献[219]。

基于树的可靠性计算(tree-based reliability calculation) 我们需要能够计算出所有节点(中心和用户)都相互连接的概率的*可靠性度量*(reliability measure)方法。下面假设树的所有元素(节点和边)的可靠性已知，计算生成树的可靠性。假设树是有根的树。每棵子树的根对应着一个状态向量。状态向量包含了该节点用于计算可靠性的所有信息。定义一系列递归关系，在给出子树状态之后可以得到树的根的状态向量。对于包含一个节点的子树，其状态是明显的。然后有根的子树用递归关系变成越来越大的子树，直到获得所有网络的状态[359]。

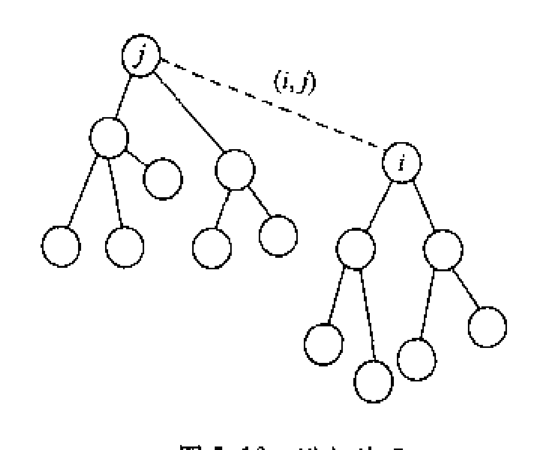


图5.12 递归关系

产生递归关系是比较机械的。该关系可以从图5.12表示的情况中得出。有2棵子树，一棵的根是节点 i，另一棵的根是节点 j。假设节点 i 的状态和节点 j 的状态都已知，我们将两个节点连接起来并使节点 j 成为节点 i 的父节点，这是需要计算节点 j 的状态。

① 译者注：如果在某个目标中，当代种群最大目标函数值和最小目标函数值的差距相对另一个目标来说较大，或者最大目标函数值相对另一个目标来说较小，则这个目标的权重系数较大。

② 译者注：对最大化问题，如果 $f_i^{(t)}(\boldsymbol{X}_k)$ 等于最大值，则归一化后适应值为1；如果 $f_i^{(t)}(\boldsymbol{X}_k)$ 等于最小值，则归一化后适应值接近0。对最小化问题，如果 $f_i^{(t)}(\boldsymbol{X}_k)$ 等于最大值，则归一化后适应值接近0；如果 $f_i^{(t)}(\boldsymbol{X}_k)$ 等于最小值，则归一化后适应值为1。

③ 译者注：经过归一化后，所有个体的归一化值都在[0,1]区间上。γ 的作用之一是防止算法收敛时出现分母为零的现象。另外，如果算法接近收敛了(即种群中所有个体的适应值都相似)，则 $d_i(\boldsymbol{X}_k)$的值约等于 $\gamma/\gamma=1$，这时基于适应值比例的选择方式表现出随机选择的特性。

假设节点失效的概率 p_i^f 和节点起作用的概率 $p_i^s(=1-p_i^f)$已知。同理分别用 l_i^f 和 l_i^s 表示边(i,j)失效和起作用的概率。同时还定义了每棵子树的状态向量：e_i 是子树中所有节点均失效的概率；o_i 是包括子树根节点在内的所有起作用节点都相互连接的概率；r_i 是子树根节点失效但其起作用节点相互连接的概率。

对于一个 n 节点并且根节点为 1 的树，可以计算该树的概率(即所有起作用节点可以相互通信的概率 $R(\boldsymbol{X})$如下：

可靠性计算过程

第 1 步：设 $r_i=0$，$o_i=p_i^s$，$e_i=p_i^f(i=1,2,\cdots,n)$，并设 $i=n$。

第 2 步：如果节点 j 是节点 i 的父节点，采用下面的递归关系来重新计算 r_i，o_i 和 e_i：

$$r'_j=r_je_i+r_ie_j+o_ie_j$$

$$o'_j=o_io_jl_i^s+o_je_i$$

$$e'_j=e_ie_j$$

第 3 步：设 $i=i-1$。如果 $i=1$，进入第 4 步，否则返回第 2 步。

第 4 步：返回 $r_1+o_1+e_1$。

选择　这里采用的选择组合了轮盘赌和最优性方法，表示如下：

第 1 步：计算每个染色体 $\boldsymbol{X}_p(p=1,2,\cdots,chr_size)$的累积概率 a_p。

第 2 步：产生[0,1]区间内的随机实数 r。

第 3 步：如果 $r\leqslant a_1$，选择第一个染色体 $\boldsymbol{X}_1$，否则选择满足 $a_{p-1}<r<a_p$ 的第 p 个染色体$(2\leqslant p\leqslant chr_size)\boldsymbol{X}_p$。

第 4 步：重复第 2 步和第 3 步 pop_size 次获得 pop_size 个染色体。

第 5 步：如果最优染色体没有被选入下一代，随机选择新种群中的一个染色体并用最优染色体替代。

杂交　采用了均匀杂交(uniform crossover)。这种杂交选择两个父代解并从中随机选择基因以形成对应子代的基因(图 5.13)。

变异　采用了交换变异(swap mutatioin)。这种变异随机选择两个点并将其内容交换，如图 5.14 所示。

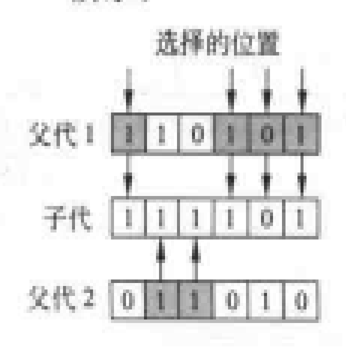


图 5.13　均匀杂交算子

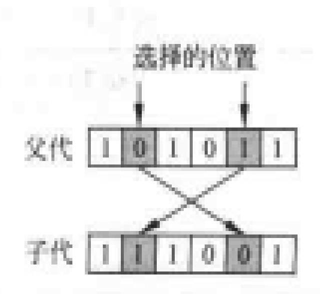


图 5.14　交换变异算子

修补染色体　由于存在每个中心最大连接容量的约束，初始种群中随机产生的染色体和由杂交产生的后代在每个中心最大连接数的意义上可能是不可行的。处理这种不可行有两种方法：惩罚方法和修补方法。对于某些组合优化问题，当不可行性难以量化时为不可行染色体提供合理的惩罚因子很困难。由于修补生成树表示的不可行染色体相对容易，因此采用了修补方法来修改不可行染色体中心的连接数。

设 $\overline{G}$ 表示染色体中最大连接数尚未检查和修改的中心的集合。如果中心 i 违反了最大连接数 g_i 的约束，这意味着染色体中这个中心的数量大于 g_i-1。于是从 $\overline{G}$ 寻找额外的中心替代它，这样就可以降低中心 i 的连接数。

5.3.2 数值例子

Gen 和 Kim 采用了下面的两个例子来说明遗传算法的性能[228]。

例 5.1　第一个问题有 4 个服务中心($n=4$)，8 个用户($m=8$)，$g_i=3$。中心 i 和中心 j 之间的权重矩阵(w_{1ij})用表 5.8 表示。中心 i 和用户 j 之间的权重矩阵(w_{2ij})用表 5.9 表示。

表 5.8　中心 i 和中心 j 之间的连接权重矩阵$[w_{1ij}]$

$$\begin{bmatrix} - & 125 & 110 & 210 \\ & - & 221 & 134 \\ & & - & 123 \\ & & & - \end{bmatrix}$$

表 5.9　中心 i 和用户 j 之间的连接权重矩阵$[w_{2ij}]$

$$\begin{bmatrix} 34 & 62 & 7 & 46 & 26 & 31 & 19 & 29 \\ 26 & 15 & 16 & 68 & 37 & 43 & 15 & 58 \\ 10 & 49 & 86 & 72 & 36 & 49 & 31 & 78 \\ 28 & 22 & 98 & 80 & 8 & 15 & 98 & 7 \end{bmatrix}$$

中心起作用的概率是 0.95，用户起作用的概率是 0.9，中心之间连线起作用的概率是 0.9，中心和用户之间连线起作用的概率是 0.85。

遗传算法的参数设置如下：$pop_size=100$，$max_gen=500$，$p_c=0.3$，$p_m=0.7$，试验进行 20 次。找到的 Pareto 最优解用图 5.15 表示。

从表 5.10 和图 5.15 中可以看出 Gen 和 Kim 的方法得到的结果比 Dengiz，Altiparmak 和 Smith 得到的结果[156,157]好。

表 5.10　Gen 等人提出的编码和基于边的编码相比较

例子	Gen 和 Kim 的方法		Dengiz、Altiparmak 和 Smith 的方法	
	ACT/s[a]	内存规模 $n+m-2$	ACT/s	内存规模 $n(n-1)/2$
5.1	5.16	10	9.37	66
5.2	33.38	34	b	630

a：ACT 为平均计算时间。

b：无法执行。

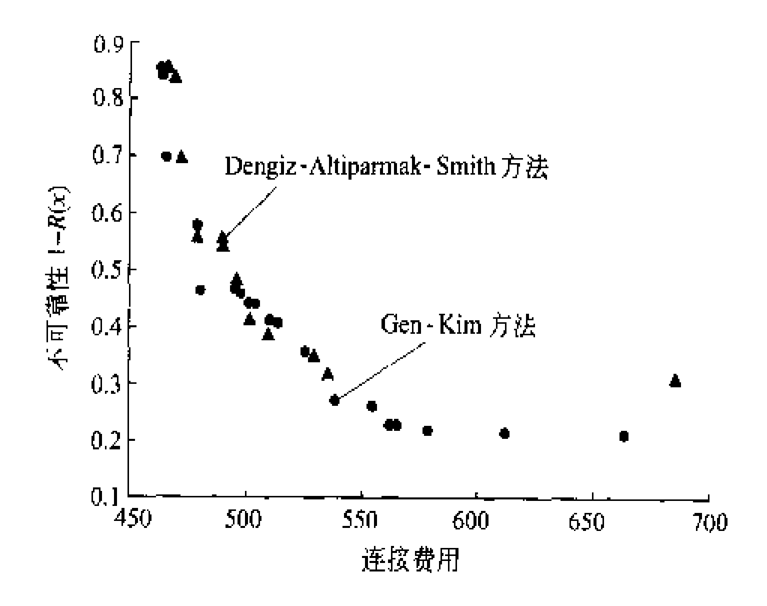


图 5.15 例 5.1 的 Pareto 解

例 5.2 第二个问题有 6 个服务中心($n=6$),30 个用户($m=30$),$g_i=10$。中心 i 和中心 j 之间的权重矩阵(w_{1ij})用表 5.11 表示。中心 i 和用户 j 之间的权重矩阵(w_{2ij})用表 5.12 表示。

中心起作用的概率是 0.95,用户起作用的概率是 0.9,中心之间连线起作用的概率是 0.9,中心和用户之间连线起作用的概率是 0.85。

表 5.11 中心 i 和中心 j 之间的连接权重矩阵[w_{1ij}]

—	125	110	210	221	105
	—	107	202	154	167
		—	175	163	231
			—	217	206
				—	157
					—

表 5.12 中心 i 和用户 j 之间的连接权重矩阵[w_{2ij}]

34	62	7	46	26	19	31	29	26	15	16	68	37	15	100	58	10	49	86	100	36	31	49	78	100	22	98	80	8	98
15	7	80	35	54	85	22	71	81	5	29	46	37	29	79	17	20	39	25	84	5	36	22	12	86	96	9	79	15	54
42	27	25	39	52	90	80	35	58	19	2	34	43	43	51	64	19	36	26	36	16	71	41	51	52	50	13	34	58	73
66	76	84	81	20	45	10	61	34	86	50	18	21	94	25	27	50	61	81	33	54	18	93	7	62	18	75	28	12	37
73	62	34	89	44	85	96	78	7	57	50	43	48	78	25	53	16	45	55	71	11	69	50	93	86	62	18	23	9	73
92	44	24	32	31	3	50	47	76	11	92	63	44	24	30	63	46	66	44	70	23	10	72	7	63	9	17	87	41	64

遗传算法的参数设置如下:$max_gen=500$,$p_c=0.3$,$p_m=0.7$,pop_size 分别设置为 50,100 和 200,试验进行 20 次。找到的 Pareto 最优解用图 5.16 表示。

Gen 和 Kim 采用 Yoon 和 Hwang 提出的 TOPSIS 方法在 Pareto 解中确定最优妥协解。TOPSIS(technique for order preference by similarity to ideal solution)的基本思想是选出的点与正理想点之间的距离要最小,而与负理想点之间的距离要最大。关于

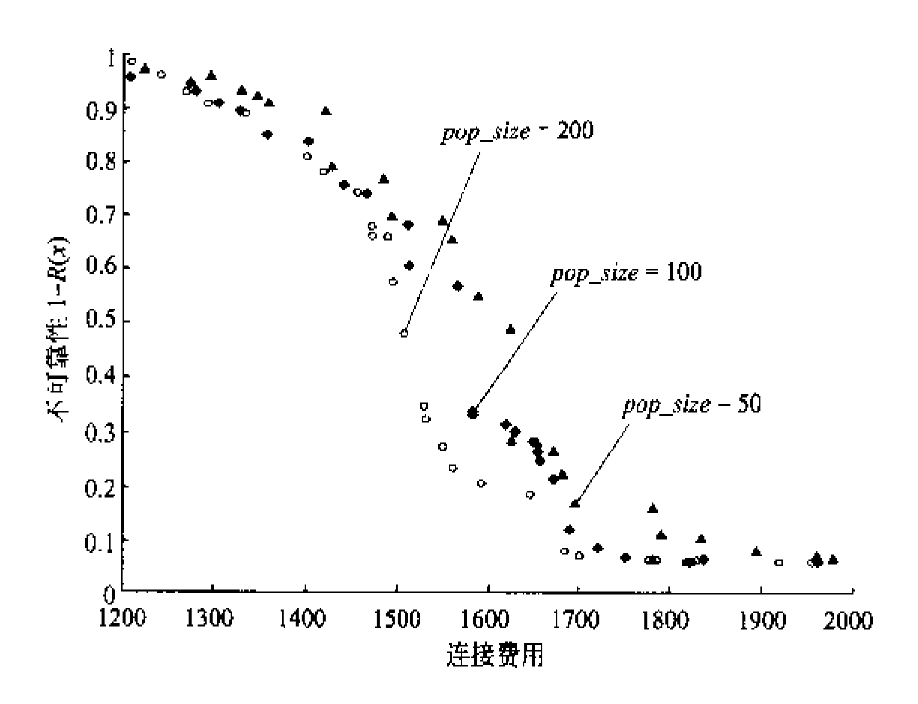


图 5.16　例 5.2 的 Pareto 解

TOPSIS 更为详细的信息参见[219,313]。图 5.17 表示了通过 TOPSIS 获得的最优妥协解。最优妥协解的染色体为

中心：6 5 6 2

用户：5 5 5 6 6 4 6 4 5 2 4 2 1 6 6 2 1 2 5 4 6 5 3 6 3 1 2 5 1 4

连接费用为 1595，网络可靠性为 0.79654。

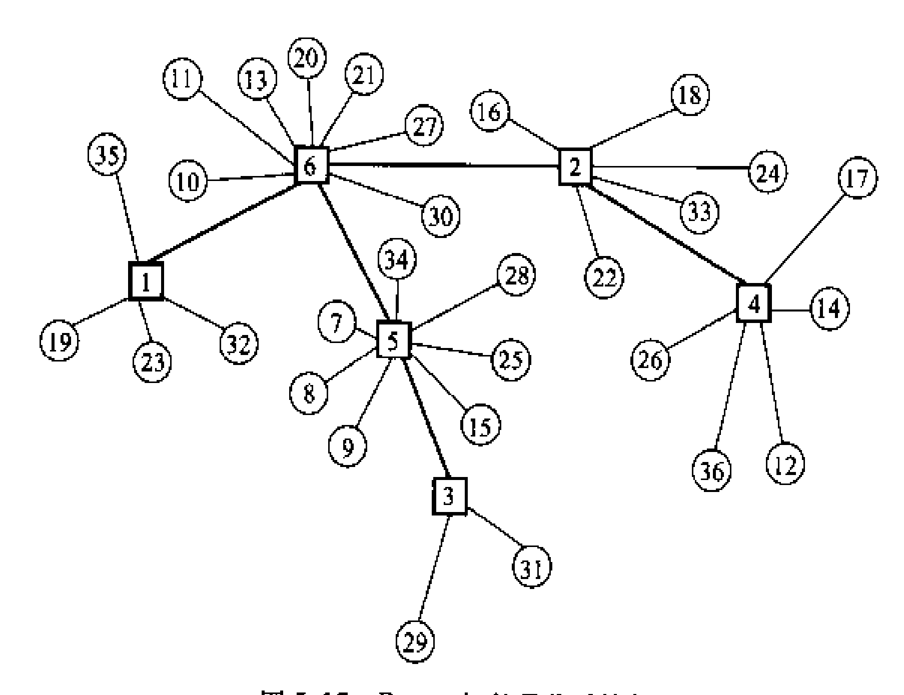


图 5.17　Pareto 解的最优妥协解

5.4 多目标可靠性设计

5.4.1 双目标可靠性设计

本小节中考虑的双目标可靠性设计问题试图最大化串联系统的可靠性，同时最小化系统的总费用。该问题是 Dhingra 给出的最优可靠性分配问题[162]的变形，可以表示为下面的非线性混合整数规划问题(nonlinear mixed-integer programming problem)[227,410]。

$$\max \quad f_1(\boldsymbol{m},\boldsymbol{x}) = \prod_{i=1}^{n}[1-(1-x_i)^{m_i}] \tag{5.7}$$

$$\min \quad f_2(\boldsymbol{m},\boldsymbol{x}) = \sum_{i=1}^{n}C(x_i)\left[m_i + \exp\left(\frac{m_i}{4}\right)\right] \tag{5.8}$$

$$\text{s.t.} \quad G_1(\boldsymbol{m}) = \sum_{i=1}^{n}w_i m_i \exp\left(\frac{m_i}{4}\right) \leqslant W_s \tag{5.9}$$

$$G_2(\boldsymbol{m}) = \sum_{i=1}^{n}v_i(m_i)^2 \leqslant V_s \tag{5.10}$$

$$1 \leqslant m_i \leqslant 10, \quad i=1,\cdots,4 \tag{5.11}$$

$$0.2 \leqslant x_i \leqslant 1-10^{-6}, \quad i=1,\cdots,4 \tag{5.12}$$

其中 m_i 是子系统 i 中冗余元件的数量，$\boldsymbol{m}=[m_1 \quad m_2 \quad \cdots \quad m_n]$。$x_i$ 是第 i 个子系统的元件可靠性(component reliability)，$\boldsymbol{x}=[x_1 \quad x_2 \quad \cdots \quad x_n]$。$f_1(\boldsymbol{m},\boldsymbol{x})$是考虑冗余元件为 $\boldsymbol{m}$ 和元件可靠性为 $\boldsymbol{x}$ 的系统的可靠性。$f_2(\boldsymbol{m},\boldsymbol{x})$是考虑冗余元件分配为 $\boldsymbol{m}$ 和元件可靠性为 $\boldsymbol{x}$ 的系统总费用。v_i 是子系统 i 中每个元素重量和体积的乘积。w_i 是子系统 i 中每个元件的重量。$C(x_i)$是子系统 i 中每个可靠性为 x_i 的元件的费用，表示如下：

$$C(x_i) = \alpha_i\left(\frac{-O_T}{\ln(x_i)}\right)^{\beta_i}, \quad i=1,\cdots,4$$

其中 α_i 和 β_i 是代表子系统 i 中元件物理特性的常数，O_T 是元件一定不能失效的操作时间。

5.4.2 遗传算法方法

Gen，Kim 和 Li 采用遗传算法来求解上述问题[230,410]。

表示和初始化 设$\boldsymbol{v}_k$表示种群中的第 k 个染色体，表示如下：

$$\boldsymbol{v}_k = [(m_{k1}, x_{k1})\ (m_{k2}, x_{k2})\ (m_{k3}, x_{k3})\ (m_{k4}, x_{k4})] \quad k=1,2,\cdots,pop_size$$

种群的初始化由染色体中的每个基因在其取值范围内随机取值来完成。

评价和选择 染色体的适应值通过下面介绍的排序方法来计算：

基于排序的评价

第1步：计算每个染色体的目标函数值。

第2步：根据染色体的目标函数值进行排序从而得到顺序 $r_i(\boldsymbol{v}_k)$。$r_i(\boldsymbol{v}_k)$表示第 k 个染色体$\boldsymbol{v}_k$的第 i 个目标函数值的次序，这个次序通过将当前种群中最好的目标函数值设为1而将最差的目标函数值设为 *pop_size* 得到。如果目标函数是最大化类型，最大目标函数值的 $r_i(\boldsymbol{v}_k)$设为1，而最小目标函数值的 $r_i(\boldsymbol{v}_k)$设为 *pop_size*。反过来，对于最小化类型，最小目标函数值的 $r_i(\boldsymbol{v}_k)$设为1，而最大目标函数值的 $r_i(\boldsymbol{v}_k)$设为 *pop_size*。

第3步：采用下面的公式计算适应值：

$$\text{eval}(\boldsymbol{v}_k) = \sum_{i=1}^{Q} r_i(\boldsymbol{v}_k)$$

其中 Q 是目标函数的数量。

采用了最优性选择(elitist selection)[219,455]。该过程的操作步骤如下：计算评价函数 $\text{eval}(\boldsymbol{v}_k)$，在父代和子代中选出相比其它染色体来说最优秀的染色体。选出的染色体数量为 *pop_size*。不允许重复选择。

杂交 采用了算术杂交(arithmetic crossover)算子，即两个染色体的线性重组[219,455]。设$\boldsymbol{v}_{k1}$和$\boldsymbol{v}_{k2}$表示第 k 代中为杂交而随机选出的两个染色体。后代可以表示为：

$$\boldsymbol{o}_{k1} = \lfloor c\boldsymbol{v}_{k1} + (1-c)\boldsymbol{v}_{k2} \rfloor$$
$$\boldsymbol{o}_{k2} = \lfloor c\boldsymbol{v}_{k2} + (1-c)\boldsymbol{v}_{k1} \rfloor$$

其中$\lfloor x \rfloor$表示小于实数 x 的最大整数，c 是[0,1]区间内的随机数。

变异 这里采用了均匀变异。对于选中的父代$\boldsymbol{v}_k$，如果其元素 $\boldsymbol{x}_{k3}$ 被随机选出进行变异，产生的子代为：

$$\boldsymbol{v}'_k = [(m_{k1}, x_{k1})\ (m_{k2}, x_{k2})\ (m_{k3}, x'_{k3})\ (m_{k4}, x_{k4})]$$

其中 x'_{k3}是在$[0.5, 1-10^{-6}]$区间上均匀概率分布的随机值。如果 m_{k3} 被选中，则返回一个可选范围内的随机整数。该算子确保遗传算法可以自由地搜索解空间，但其缺点是后期的分散性。

数值例子和结果 Gen，Kim 和 Li 采用 Dhingra 的例子来测试他们的算法。输入参见表5.13。参数设置为：$pop_size=200$，$p_c=0.4$，$p_m=0.6$，$max_gen=1000$。得到的 Pareto 解表示在表5.14中。

表 5.13 可靠性分配问题的常数系数

子系统数量	4
W_s 的限制	500.0
V_s 的限制	250.0
操作时间 t_0	1000 h

子系统	$10^5\alpha_j$	β_j	v_j	w_j
1	1.0	1.5	1	6
2	2.3	1.5	2	6
3	0.3	1.5	3	8
4	2.3	1.5	2	7

图 5.14 Pareto 解

$R(m,x)$	$C(m,x)$	$G_1(m,x)$	$G_2(m,x)$	染色体
0.998395	398.559	396.25	217.00	[(0.777820, 5) (0.697710, 6) (0.916213,4) (0.748629, 6)]
0.995562	313.737	391.43	244.00	[(0.791324, 6) (0.705309, 5) (0.682558,6) (0.743228, 5)]
0.992422	253.985	369.69	235.00	[(0.764636, 4) (0.607286, 6) (0.801730,5) (0.713641, 6)]
0.989157	206.800	364.25	206.00	[(0.670458, 6) (0.689409, 5) (0.832996,4) (0.574555, 6)]
0.987813	186.650	386.00	244.00	[(0.694460, 5) (0.649614, 6) (0.769595,5) (0.561921, 6)]
0.972312	136.093	386.00	244.00	[(0.602327, 5) (0.576630, 6) (0.729846,5) (0.529785, 6)]

5.4.3 混合遗传算法方法

Gen 和 Kim 提出了混合遗传算法来求解双目标可靠性设计问题[229]。Hooke-Jeeves 方法(Hooke-Jeeves method)是一种直接搜索(direct search)方法，将其嵌入遗传算法的主循环中并对每个新产生的子代使用，将子代移动到最近的局部最优点上。混合遗传算法(hybrid genetic algorithm)应用在式(5.7)～式(5.12)表示的问题上。

表示和初始化 由于有两种类型的决策变量，因此采用了下面的染色体表示方式：

$$\boldsymbol{v}_k = [(m_{k1}, x_{k1})(m_{k2}, x_{k2})(m_{k3}, x_{k3})(m_{k4}, x_{k4})] \quad k = 1,2,\cdots,pop_size$$

其中 m_{ki} 是对应着冗余元件数量的整数，x_{ki} 是对应着元件可靠性水平的实数。种群的初始化由染色体中的每个基因在其取值范围内随机取值来完成。

评价和选择 适应值函数采用权重和的形式来将多个目标合并成为单个目标。权重系数由当前代中目标函数值的分散程度决定。具体的过程描述如下：

评价过程

第 1 步：计算目标函数值[$f_i(\boldsymbol{v}_k)$, $i=1,2$, $k=1,2,\cdots,chr_size$]，其中 chr_size 是父代和子代的总数量。

第 2 步：将目标函数值标准化如下：

$$\hat{f}_i(\boldsymbol{v}_k) = \frac{f_i(\boldsymbol{v}_k) - \bar{f}_i}{\sigma_i}, \quad i - 1,2, \quad k = 1,2,\cdots,chr_size$$

其中 $\bar{f}_i$ 是整个种群中第 i 个目标函数的平均值，σ_i 是第 i 个目标函数的标准差。

第 3 步：用整个种群的平均值 h 和标准差 d 来计算比例变换值 $c_i(\boldsymbol{v}_k)$ 如下：

$$c_i(\boldsymbol{v}_k) = d\hat{f}_i(\boldsymbol{v}_k) + h, \quad i = 1,2 \quad k = 1,2,\cdots,chr_size$$

第 4 步：确定每个目标函数最大和最小的标准化值：

$$f_i^{\min(t)} = \min\left\{ (f_i^{\min(t-1)}, f_i^{(t)}(\boldsymbol{v}_k)) \middle| k \in [1,chr_size] \right\}$$

$$f_i^{\max(t)} = \max\left\{ (f_i^{\max(t-1)}, f_i^{(t)}(\boldsymbol{v}_k)) \middle| k \in [1,chr_size] \right\}, \quad i = 1,2$$

其中 $f_i^{\min(t)}$ 和 $f_i^{\max(t)}$ 分别是第 i 个目标在第 t 代的最小目标函数值和最大目标函数值。$f_i^{(t)}(\boldsymbol{v}_k)$ 是第 k 个染色体第 t 代时第 i 个目标函数值。

第 5 步：计算权重系数如下：

$$\lambda_i = \frac{\delta_i}{\sum_{i=1}^{2}\delta_i}, \quad i = 1,2$$

其中

$$\delta_i = \frac{f_i^{\max(t)} - f_i^{\min(t)}}{f_i^{\max(t)}}$$

第 6 步：对每个染色体计算适应值如下：

$$\mathrm{eval}(\boldsymbol{v}_k) = \sum_{i=1}^{2}\lambda_i c_i(\boldsymbol{v}_k), \quad k = 1,2,\cdots,chr_size$$

杂交和变异　采用了算术杂交(arithmetic crossover)算子。设 $\boldsymbol{v}_{k1}$ 和 $\boldsymbol{v}_{k2}$ 表示第 k 代中为杂交而随机选出的两个染色体。后代可以表示为

$$\boldsymbol{o}_{k1} = \lfloor c\boldsymbol{v}_{k1} + (1 \quad c)\boldsymbol{v}_{k2} \rfloor$$

$$\boldsymbol{o}_{k2} = \lfloor c\boldsymbol{v}_{k2} + (1 - c)\boldsymbol{v}_{k1} \rfloor$$

其中 $\lfloor x \rfloor$ 表示小于实数 x 的最大整数，c 是[0,1]区间内的随机数。

对于选中的父代 $\boldsymbol{v}_k$，假设其元素 $\boldsymbol{x}_{k3}$ 被随机选出进行变异，产生的子代为

$$\boldsymbol{v}'_k = [(m_{k1}, x_{k1})(m_{k2}, x_{k2})(m_{k3}, x'_{k3})(m_{k4}, x_{k4})]$$

其中 x'_{k3} 是在 $[0.5, 1-10^{-6}]$ 区间上均匀概率分布的随机值。如果 m_{k3} 被选中，则返回一个可选范围内的随机整数。

Hooke-Jeeves 方法(Hooke-Jeeves method)　将该方法作为局部搜索方法来改善遗传算法的能力[526]。Hooke-Jeeves 方法是顺序化方法，每一步包含两种类型的移动：探索

(exploratory)和模仿(pattern)。探索移动检查待优化函数的局部性质并试图定位任何可能出现倾斜山谷的方向。模仿移动采用了探索移动获得的信息在山谷中快速移动。Gen 和 Kim 采用了下面介绍的 Hooke-Jeeves 方法来求解双目标可靠性设计问题[227]。

Hooke-Jeeves 方法过程

第 1 步：从一个染色体$\boldsymbol{v}$开始，该染色体称作开始基本点，给出每个坐标方向 $\boldsymbol{d}_i(i=1,2,\cdots,n)$上的步长 δ_i。

第 2 步：计算 $f_s(\boldsymbol{v})$，其中如果 $s=1$，则 $f_s(\boldsymbol{v})=-f_1(\boldsymbol{v})$，否则 $f_s(\boldsymbol{v})=f_2(\boldsymbol{v})$。$s$ 是$[1,2]$上的随机整数。设 $i=1,\boldsymbol{u}_0=\boldsymbol{v}$，开始第 3 步描述的探索移动。①

第 3 步：每个变量在当前临时基本点 $\boldsymbol{u}_{i-1}$的周围进行扰动以获得一个新的临时基本点

$$\boldsymbol{u}_i=\begin{cases}\boldsymbol{u}_{i-1}+\delta_i\boldsymbol{d}_i, & F^+<F\\ \boldsymbol{u}_{i-1}-\delta_i\boldsymbol{d}_i & F^-<F<F^+\\ \boldsymbol{u}_{i-1}, & \min\{F^-,F^+\}>F\end{cases} \tag{5.13}$$②

其中点 $\boldsymbol{u}_i$ 表示从基本点 $\boldsymbol{u}_{i-1}$通过扰动$\boldsymbol{v}$ 的第 i 个元素获得的临时基本点。$F^+=f_s(\boldsymbol{u}_{i-1}+\delta_i\boldsymbol{d}_i)$，$F^-=f_s(\boldsymbol{u}_{i-1}-\delta_i\boldsymbol{d}_i)$，$F=f_s(\boldsymbol{u}_{i-1})$。寻找新临时基本点的过程持续到所有方向都进行了扰动得到 $\boldsymbol{u}_n$ 为止。

第 4 步：如果点 $\boldsymbol{u}_n$ 与$\boldsymbol{v}$ 一样，减小步长 δ_i(除以 2)，设 $i=1$，返回第 3 步。如果 $\boldsymbol{u}_n$ 与$\boldsymbol{v}$ 不同，则产生一个新的基本点$\boldsymbol{v}'=\boldsymbol{u}_n$，进入第 5 步。

第 5 步：在基本点$\boldsymbol{v}$ 和$\boldsymbol{v}'$的帮助下，建立模仿移动方向 $\boldsymbol{s}=\boldsymbol{v}'-\boldsymbol{v}$，产生新的染色体

$$\boldsymbol{v}_{\text{new}}=\boldsymbol{v}'+\boldsymbol{s} \tag{5.14}$$③

上述过程在选择操作之后采用，目的是在其进行杂交和变异之前将其移动到局部最优解上。

数值例子和结果 Gen，Kim 和 Li 采用 Dhingra 给出的最优可靠性分配问题[162]的变形作为数值例子，该问题是非线性混合整数规划问题。该问题包括两个目标：最大化可靠性和最小化费用。输入参见表 5.13。

对于这个问题，参数设置为：$h=50$，$d=10$，$pop_size=40$，$p_c=0.5$，$p_m=0.1$，$max_gen=2000$。得到的 Pareto 解表示在表 5.18 中。多次试验得到的结果表明：混合遗传算法的性能比不采用 Hooke-Jeeves 方法的简单遗传算法性能好[227,229]。

---

① 将最大化目标函数转换为最小化目标函数。

② 译者注：如果沿着 $\boldsymbol{d}_i$ 的正方向移动减少目标函数值，则新基本点向着 $\boldsymbol{d}_i$ 的正方向移动；如果沿着 $\boldsymbol{d}_i$ 的负方向移动减少目标函数值并且沿着 $\boldsymbol{d}_i$ 的正方向移动增加目标函数值，则新基本点向着 $\boldsymbol{d}_i$ 的负方向移动；如果沿着 $\boldsymbol{d}_i$ 的正负方向移动均增大目标函数值，则基本点保持不变。

③ 译者注：式(5.14)的含义是在找到下降方向 $\boldsymbol{s}$ 以后从$\boldsymbol{v}'$继续沿着 $\boldsymbol{s}$ 向前移动基本点以获得快速的收敛效果。

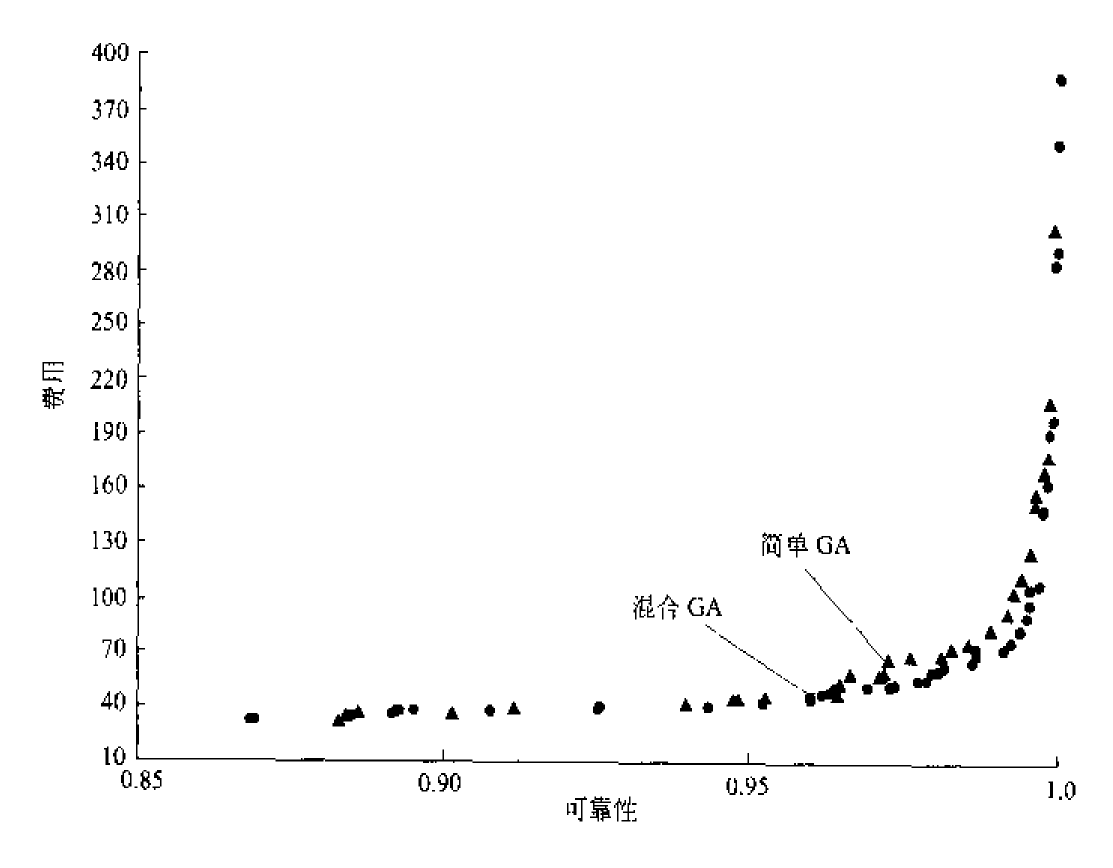


图 5.18　Pareto 解

5.4.4　带有模糊目标的可靠性设计

Gen, Ida 和 Kim 提出了解决带有模糊目标的可靠性设计问题的遗传算法[224]。该问题是 Dhingra 给出的最优可靠性分配问题的变形，表示为下面的非线性混合整数规划问题：

$$\max \quad R(\boldsymbol{m},\boldsymbol{x}) = \prod_{i=1}^{4}[1-(1-x_i)^{m_i}] \gtrsim b_1 \tag{5.15}$$

$$\min \quad C(\boldsymbol{m},\boldsymbol{x}) = \sum_{i=1}^{4} C(x_i)\left[m_i + \exp\left(\frac{m_i}{4}\right)\right] \lesssim b_2 \tag{5.16}$$

$$\min \quad W(\boldsymbol{m}) = \sum_{i=1}^{4} w_i m_i \exp\left(\frac{m_i}{4}\right) \lesssim b_3 \tag{5.17}$$

$$\text{s.t.} \quad G_1(\boldsymbol{m}) = \sum_{i=1}^{4} v_i (\boldsymbol{m}_i)^2 \leqslant 250 \tag{5.18}$$

$$1 \leqslant m_i \leqslant 10, \quad m_i \text{ 为整数}, \quad i = 1,\cdots,4 \tag{5.19}$$

$$0.2 \leqslant x_i \leqslant 1-10^{-6}, \quad x_i \text{ 为实数}, \quad i = 1,\cdots,4 \tag{5.20}$$

其中 v_i 是阶段 i 中每个元素重量和体积的乘积。w_i 是阶段 i 中每个元件的重量。$C(x_i)$

是节点 i 中每个可靠性为 x_i 的元件的费用，表示如下：

$$C(x_i) = \alpha_i\left(\frac{-O_T}{\ln(x_i)}\right)^{\beta_i}, \quad i = 1,\cdots,4$$

其中 α_i 和 β_i 是代表子系统 i 中元件物理特性的常数，O_T 是元件一定不能失效的操作时间。F_1 等于 $b_1-t_1^L$，F_2 等于 $b_2+t_2^R$，F_3 等于 $b_3+t_3^R$。符号$\gtrsim$（模糊大于）指的是决策者甚至在结果比期望水平小的程度在容差限制 t_k^L 之内依然满意。符号$\lesssim$（模糊小于）指的是决策者甚至在结果比期望水平大的程度在容差限制 t_k^R 之内依然满意。

根据 Tiwari，Dharmar 和 Dao 的工作，该问题可以转换为下面的非线性混合整数规划问题[122,622]：

$$\max \quad q_R\lambda_R + q_C\lambda_C + q_W\lambda_W \tag{5.21}$$

$$\text{s.t.} \quad \lambda_R = \mu_R(R(\boldsymbol{m},\boldsymbol{x})) \tag{5.22}$$

$$\lambda_C = \mu_C(R(\boldsymbol{m},\boldsymbol{x})) \tag{5.23}$$

$$\lambda_W = \mu_W(R(\boldsymbol{m},\boldsymbol{x})) \tag{5.24}$$

$$G_1(\boldsymbol{m}) = \sum_{i=1}^{4} v_i m_i^2 \leqslant 250 \tag{5.25}$$

$$0 \leqslant \lambda_R \leqslant 1,\ 0 \leqslant \lambda_C \leqslant 1,\ 0 \leqslant \lambda_W \leqslant 1 \tag{5.26}$$

其中 q_R，q_C 和 q_W 由决策者指定的权重系数[622]。模糊目标的隶属度函数可以定义如下：

$$\mu_R(r(\boldsymbol{m},\boldsymbol{x})) = \begin{cases} 0, & R(\boldsymbol{m},\boldsymbol{x}) < b_1 - t_1^L \\ \dfrac{R(\boldsymbol{m},\boldsymbol{x}) - (b_1 - t_1^L)}{t_1^L}, & b_1 - t_1^L \leqslant R(\boldsymbol{m},\boldsymbol{x}) \leqslant b_1 \\ 1, & R(\boldsymbol{m},\boldsymbol{x}) > b_1 \end{cases}$$

$$\mu_C(C(\boldsymbol{m},\boldsymbol{x})) = \begin{cases} 1, & C(\boldsymbol{m},\boldsymbol{x}) < b_2 \\ 1 - \dfrac{C(\boldsymbol{m},\boldsymbol{x}) - b_2}{t_2^R}, & b_2 \leqslant C(\boldsymbol{m},\boldsymbol{x}) \leqslant b_2 + t_2^R \\ 0, & C(\boldsymbol{m},\boldsymbol{x}) > b_2 + t_2^R \end{cases}$$

$$\mu_W(W(\boldsymbol{m},\boldsymbol{x})) = \begin{cases} 1, & W(\boldsymbol{m},\boldsymbol{x}) < b_3 \\ 1 - \dfrac{W(\boldsymbol{m},\boldsymbol{x}) - b_3}{t_3^R}, & b_3 \leqslant W(\boldsymbol{m},\boldsymbol{x}) \leqslant b_3 + t_3^R \\ 0, & W(\boldsymbol{m},\boldsymbol{x}) > b_3 + t_3^R \end{cases}$$

该问题的常数系数由表 5.15 给出[162]。

表 5.15　带有模糊目标的可靠性设计的常数系数

子系统	$10^5\alpha_j$	β_j	v_j	w_j
子系统数量		4		
F1 的限制		0.75		
F2 的限制		400.0		
F3 的限制		500.0		
V 的上界限制		250.0		
操作时间 t_0		1000 小时		
1	1.0	1.5	1	6
2	2.3	1.5	2	6
3	0.3	1.5	3	8
4	2.3	1.5	2	7

表示和初始化　设 v_k 表示种群中的染色体。染色体表示如下：

$$\boldsymbol{v}_k=[(m_{k1},x_{k1})\ (m_{k2},x_{k2})\ (m_{k3},x_{k3})\ (m_{k4},x_{k4})],\quad k=1,2,\cdots,pop_size$$

其中 m_{ki} 对应着冗余元件的数量，x_{ki} 对应着元件可靠性的水平。种群的初始化由染色体中的每个基因在其取值范围内随机取值来完成。

染色体的评价　适应值函数采用下面的形式：

$$\text{eval}(\boldsymbol{v}_k)=q_R\lambda_R+q_C\lambda_C+q_W\lambda_W,\quad k=1,2,\cdots,pop_size$$

最优染色体 $\boldsymbol{v}^*$ 在每代中根据下式选出：

$$\boldsymbol{v}^*=\max\{\text{eval}(\boldsymbol{v}_k),\quad k=1,2,\cdots,pop_size\}$$

杂交　采用了算术杂交(arithmetic crossover)算子。设 $\boldsymbol{v}_{k1}$ 和 $\boldsymbol{v}_{k2}$ 表示第 k 代中为杂交而随机选出的两个染色体。后代可以表示为

$$\boldsymbol{o}_{k1}=\lfloor c\,\boldsymbol{v}_{k1}+(1-c)\,\boldsymbol{v}_{k2}\rfloor$$

$$\boldsymbol{o}_{k2}=\lfloor c\,\boldsymbol{v}_{k2}+(1-c)\,\boldsymbol{v}_{k1}\rfloor$$

其中 c 是[0,1]区间内的随机数。对于染色体中的整数变量，算术杂交的结果可能导致实数。我们简单地将其整数部分作为最终的基因。

变异　采用了均匀变异。对于选中的父代 $\boldsymbol{v}_k$，如果其元素 x_{k3} 被随机选出进行变异，产生的子代为

$$\boldsymbol{v}'_k=[(m_{k1},x_{k1})\ (m_{k2},x_{k2})\ (m_{k3},x'_{k3})\ (m_{k4},x_{k4})]$$

其中 x'_{k3} 是在$[0.5,\ 1-10^{-6}]$区间上均匀概率分布的随机值。如果 m_{k3} 被选中，则返回一个可选范围内的随机整数。该算子确保遗传算法可以始终自由地搜索解空间，但其缺点是后期的分散性。

选择　这里采用的选择组合了轮盘赌和最优性方法。采用轮盘赌作为基本方法来产生新的一代；采用最优性方法在下一代中保持最优染色体并克服采样的随机误差。

数值例子和结果　参数设置为：$pop_size=20$，$p_c=0.4$，$p_m=0.1$，$T=2000$，$t_1^L=0.25$，$t_2^R=300$，$t_3^R=360$。选择的目标之间相对权重为(0.5，0.25，0.25)。

在1923代找到的最优解的目标函数值为：$R(\boldsymbol{m},\boldsymbol{x})=0.982335$，$C(\boldsymbol{m},\boldsymbol{x})=135.001259$，$W(\boldsymbol{m},\boldsymbol{x})=147.048541$，$(\boldsymbol{m},\boldsymbol{x})=[(3,0.853070)(3,0.821421)(2,0.939903)(3,0.825620)]$。文献[158]给出的最优解为 $R(\boldsymbol{m},\boldsymbol{x})=0.94478$，$C(\boldsymbol{m},\boldsymbol{x})=104.472$，$W(\boldsymbol{m},\boldsymbol{x})=128.727$，$(\boldsymbol{m},\boldsymbol{x})=[(2,0.86504)(3,0.81814)(2,0.84253)(3,0.80645)]$。

第6章 调度问题

6.1 引言

广义地讲，调度问题考虑的是随着时间的变化，如何调度有限的资源在执行任务的同时满足特定约束。资源可能在本质上是很不相同的：人力、金钱、机器、工具、材料、能源等等。任务也可以有不同的解释，从制造系统的机器划分到计算机系统的信息处理。一项任务通常可以用下面的因素来表示特征：完成时间、预期时间、相对紧急权重、处理时间和资源消耗。同时，一组反映任务之间先后约束的结构可以用不同的方式定义。另外还可以考虑度量调度性能好坏的不同判据。

调度问题几乎在现实环境（特别是工业工程领域）中无处不在。许多制造工业提出的调度问题从本质上讲非常复杂，难以用传统优化方法求解。通常这些难于求解的问题都表示为满足非常复杂约束的组合优化（combinatorial optimization）问题，这些问题带有有限数量的可行解。

这些问题属于 NP-难问题。因此引起了用遗传算法处理这些问题的兴趣。在下面几节中将说明如何用遗传算法来求解这样的问题，包括作业车间调度、群体作业调度、并行机器调度、资源约束的项目调度和多处理器调度。

6.2 作业车间调度

在作业车间调度（job-shop scheduling）问题中，给定一个作业（job）的集合和一个机器（machine）的集合。每台机器同一时间最多可以加工一个作业。每个作业包括一系列工序（operation），每个工序需要在特定的机器上不间断地加工事先给定的时间。研究的目的是确定一个调度，该调度将每个工序分配到对应机器的某个时间段，同时最小化完成所有作业所需的最小加工持续时间[37]。

该问题是最有名的复杂组合优化问题之一。在过去的 30 年中，这个问题得到了众多研究人员的重视，并提出了许多求解方法。这些方法包括从简单而快速的分配规则到复杂的分支定界算法。然而，随着计算速度的迅速增长和对调度有效性要求的显著增加，越来越强调在消耗某些额外计算费用的基础上获得更好的调度方案。

6.2.1 基本方法

作业车间调度问题的本质是建立工序在每台机器上的排列，该排列可以满足用于最小化生产时间的先后约束。如果能够得到这样的排列，就可以很轻松地通过依赖于问题的过程得到问题的解。将遗传算法应用于作业车间调度问题的一般方法为：

1. 采用遗传算法来进化适当的排列。
2. 采用启发式方法来根据排列构造对应的解。

由于遗传算法不善于在最优解附近进行精细搜索，人们提出了许多不同的与其他算法相混合的方法来克服这个缺点。这种针对作业车间问题的混合方法可以分成如下三类[147,533]：(1)适应性遗传算子、(2)以启发式为特征的遗传算子，(3)混合遗传算法。第一种方法修改或发明遗传算子来满足给定的编码表示的特性。第二种方法通过传统启发式方法来创建新的遗传算子。第三种方法在遗传算法的主循环中合适的地方混入传统启发式方法。

6.2.2 编码

一般将杂交算子看作主要的遗传算子，遗传算法的性能在很大程度上受到其采用的杂交算子性能的影响。从概念上讲杂交算子同时在两个染色体上执行，通过组合两个染色体的特性来产生后代。实现杂交的简单方法可以采用有名的单点或两点杂交方法，该方法在位串编码方面表现良好。对于许多复杂问题，很难将解用位串编码(bit string encodings)的方式来表示。因此提出了许多非位串编码(特别是各种字母串编码(literal string encodings))，同时也产生了许多与之相匹配的新的遗传算子。新产生的杂交方法可以从本质上看作单点或两点杂交针对非位串编码的修订版，它结合了修补过程，从而避免产生非法解或不可行解。

为了便于解释，我们首先定义两种类型的字母串：纯的和一般的。纯字母串编码(pure literal string encoding)包括不同的符号；而一般字母串编码(general literal string encoding)则允许一个符号重复事先指定的次数。换句话说，纯字母串编码禁止重复，而一般字母串编码则允许相同符号并存。

在过去的10年里提出了下面9种作业车间调度问题的表示方法[314]。

1. 基于工序的表示
2. 基于作业的表示
3. 基于偏好列表的表示
4. 基于作业对关系的表示
5. 基于优先规则的表示
6. 基于非连接图的表示

7. 基于完成时间的表示

8. 基于机器的表示

9. 随机键表示

在这9种表示方法中，基于作业的表示和基于机器的表示属于纯字母串表示类型，基于工序的表示、基于偏好列表的表示、基于优先规则的表示属于一般字母串表示类型。值得指出，在许多研究工作中成功地采用了基于偏好列表的表示。该方法包括若干子串，一个子串是一个对应一台机器工序顺序的纯字母串。这种编码在实践中常用的方法是将遗传算子应用于每个子串上。这样多次应用遗传算子就可以基本上把该编码看作纯字母串。因此纯字母串编码的许多遗传算子都可以直接应用于该编码上。

据我们所知，作业车间调度问题存在两种类型的次序关系：(1)每台机器的工序顺序(operation sequence)，(2)一个作业工序间的先后约束(precedence constraints)。第一种关系必须由求解方法来确定，而第二种关系则必须由调度来维持。因此，采用了不同的过程来处理次序关系。其中一种就是在编码中同时保留关于工序顺序和先后约束的信息。对于这种情况，对于字母排列编码的所有杂交方法都不能直接应用。一个染色体可能由于某些先后顺序不能满足而不可行，也可能由于某些符号重复次数不等于事先指定的数而非法。对于纯字母串编码，非法性意味着某些符号重复了超过一次，某些符号则丢失了。对于一般字母串编码非法性意味着某些符号重复的次数超过必要值，而其他的则少于必要值。当针对这种类型编码设计新杂交算子时必须对于处理不可行和非法性给予特别的考虑，从而不会破坏先后关系。在字母串编码中，基于工序的编码是惟一在一个染色体中同时保持工序顺序和先后约束信息的方法。

另一种情况是仅对关于工序顺序的信息进行编码，这就需要采用特殊的解码器或调度产生过程来得到先后约束。在这种情况下，染色体是一种类型的字母排列，仅考虑不要让杂交算子产生非法后代。在字母串编码中，基于偏好列表的表示、基于作业的表示和基于机器的表示属于这种类型。

最后一种情况是两种次序关系都没有进行编码，而是将某种行会信息(guild information)编码进入染色体。这种编码是字母串的纯排列。虽然一些基因的重复导致非法性(和上面的情况一样)，基因的次序与作业中工序的次序没有直接的关系。基于优先规则的编码属于这种类型。

6.2.3 适应性遗传算子

在过去的20年中，对于字母排列编码提出了不同的杂交算子，比如部分映射杂交(PMX)、次序杂交(OX)、基于为止的杂交、基于次序的杂交和循环杂交(CX)。在为字母排列编码(literal permutation encoding)设计杂交算子时要注意两个基本的考虑：

1. 杂交时变化尽量少，目的是从父代继承尽可能多的信息。两点杂交的所有变形都

属于这一类。

2. 杂交时变化尽量大，目的是探索排列的新模式并由此增强搜索能力[115]。均匀杂交的所有变形都属于这一类。

部分映射杂交(partially mapped crossover，PMX) 部分映射杂交由 Goldberg 和 Lingle 提出[252]。该方法可以看作两点杂交的变形，它通过引入一种特殊的修补过程来处理可能的非法现象。PMX 的主要步骤如下：

第 1 步：在串上随机选择两个分割点。由两个分割点定义的子串称作映射片断(mapping sections)。

第 2 步：在两个父代间交换子串从而产生原型后代。

第 3 步：确定两个映射片断间的映射关系。

第 4 步：采用映射关系使后代合法化。

次序杂交(order crossover，OX) 次序杂交由 Davis 提出[146]。该方法可以看作 PMX 的一种变形，它采用了不同的修补过程。OX 的主要步骤如下：

第 1 步：随机从一个父代中选择一条子串。

第 2 步：通过将子串复制到与其父代相对应的位置上产生原型子代。

第 3 步：将第二个父代中所有子串中已有的符号全部去掉，余下的顺序包含了原型子代需要的符号。

第 4 步：根据从左到右的顺序将符号放入原型子代中余下的空位中，这样就产生了一个子代。

基于位置的杂交(position-based crossover) 基于位置的杂交由 Syswerda 提出[603]。该方法基本上是针对字母排列编码并结合了修改过程的均匀杂交。针对位串编码的均匀杂交由 Syswerda 提出[603]。该方法首先产生一个随机的掩码，然后根据掩码交换父代中对应的基因。杂交掩码就是与染色体长度相同的二进制串。掩码中每位的奇偶性决定了后代中该位从哪个父代所对应的位继承。由于均匀杂交对于字母排列编码将产生非法解，基于位置的杂交采用了一种修补过程来解决非法现象。基于位置杂交的主要步骤如下：

第 1 步：从一个父代中随机选择位置的集合。

第 2 步：将父代中这些位置的符号复制到原型子代中对应的位置上。

第 3 步：在第二个父代中删除已经被选择的符号。余下的顺序包含了原型子代需要的符号。

第 4 步：根据顺序从左到右将符号放入原型子代余下的空位中，这样就产生了一个子代。

基于次序的杂交(order-based crossover) 基于次序的杂交也是由 Syswerda 提出的[603]。该方法与基于位置的杂交有少许不同。它将一个父代中选出的位置上的符号次序加在另一个父代对应的位置上。

循环杂交(cycle crossover，CX) 循环杂交由 Oliver，Smith 和 Holland 提出[490]。类

似于基于位置的杂交，该方法从一个父代中提取某些符号，余下的符号从另一个父代中提取。区别在于从第一个父代中提取的符号不是随机选择的，而是选择那些能够根据父代之间对应位置构成一个循环的符号。CX 的步骤如下：

第 1 步：寻找根据父代之间对应位置构成的一个循环。

第 2 步：从一个父代中将循环中的符号保持其位置复制到子代中。

第 3 步：从另一个父代中删除那些属于循环的符号，余下的部分就构成了子代中所需的符号。

第 4 步：用余下的符号填充子代。

线性次序杂交(linear order crossover，LOX) Falkenauer 和 Bouffoix 提出了一种次序杂交的改进方法，称作线性次序杂交[184]。次序杂交倾向于传递基因间的相对位置，而不是绝对位置。由于次序杂交是针对旅行推销员问题(TSP)设计的，引起该方法认为染色体是循环的。在作业车间问题中，染色体不能看作循环。基于这个原因，Falkenauer 和 Bouffoix 开发了一种 OX 的变形，称作线性次序杂交(LOX)。该方法线性地而不是循环地考虑染色体。LOX 的步骤如下：

第 1 步：从父代中随机选择子序列。

第 2 步：从父代 p_1 中删除 $sublist_2$，这样就产生了一些"洞"。然后从两端向中心移动这些洞，直到它们到达交叉部分为止。类似地，从父代 p_2 中删除 $sublist_1$ 并将洞移动到交叉部分为止。

第 3 步：将 $sublist_1$ 插入 p_2 的洞中以构成后代 o_1，将 $sublist_2$ 插入 p_1 的洞中以构成后代 o_2。

该杂交算子可以尽可能多地保留基因间的相对位置和基因与父代末端的绝对位置。末端对应着高或者低的优先工序。

子顺序交换杂交(subsequence exchange crossover) Kobayashi，Ono 和 Yamamura 受到 Brady[74] 以及 Mühlenbein，Schlouter 和 Kraemer[470] 针对 TSP 类似思想的影响提出了子顺序交换杂交方法[371]。该方法采用作业顺序矩阵作为编码方式。对于 n 个作业 m 个机器的问题，编码是 $m \times n$ 矩阵，其中每行对应着每台机器的工序顺序。子顺序定义为一个作业的集合，两个父代中的这些作业都在一台机器上连续进行加工，但加工的次序不一定相同。

第 1 步：确定父代之间每台机器的子顺序。

第 2 步：在父代间交换这些机器的子顺序来生成子代。

由于采用作业顺序矩阵编码方式难以在初始种群和杂交后代中维持工序之间的先后关系，因此采用了 Giffler 和 Thompson 算法来精细调整每台机器上的作业次序。这样就解决了不可行的问题，同时也把后代转换为活动调度。

基于作业的次序杂交(job-based order crossover) Ono，Yamamura 和 Kobayashi 放松了对所有子顺序中作业连续进行加工的要求，进而提出了对子序列交换杂交的变形方

法，称作基于作业次序的杂交[493]。基于作业次序的杂交也是为作业顺序矩阵编码方式而设计的。

第1步：从父代中确定作业集合，每个机器对应一个集合。

第2步：对于每台机器，从第一个父代中将选出的作业复制到第一个子代中对应的位置上。用相同的方法处理第二个子代。

第3步：根据第二个父代中未选中作业从左到右的顺序填充第一个子代中未确定的作业。用相同的方法处理第二个子代。

部分调度交换杂交(partial schedule exchange crossover) Gen, Tsujimura 和 Kubota 针对基于工序的编码方式提出了部分调度交换杂交[240]。他们将部分调度看作自然的积木块，并希望利用杂交在子代中维持积木块，这与 Holland 描述的方式相同[303]。

第1步：在一个父代中随机确定一个部分调度，由此确定另一个父代中的部分调度。

第2步：交换部分调度，从而产生原型子代。

第3步：确定原型子代中缺乏的和多余的基因。

第4步：通过删除多余基因和添加缺乏基因来使后代合法化。

如果一个调度的一部分在头和尾部都出现了相同的作业，就构成了部分调度。

子串交换杂交(substring exchange crossover) Cheng, Gen 和 Tsujimura 提出了另一种类型的部分调度交换杂交，称作子串交换杂交[115]。该方法可以看作两点杂交对于一般字母串编码的适应性改进。

第1步：随机在串中选择两个交叉点。交换两个父代由两个交叉点确定的子串，从而产生两个原型子代。

第2步：通过比较两个子串来确定每个原型子代中缺乏的和多余的基因。

第3步：通过随机将多余的基因替换为缺乏的基因来使原型子代合法化。

变异 对排列表示进行变异操作相对比较容易。在过去的10年里提出了若干针对排列表示的变异算子，包括反转变异、插入变异、位移变异、互反变异和移动变异[219]。*反转变异*(inversion mutation)在染色体中随机寻找两个位置，然后将这两个位置之间的子串进行反转。*插入变异*(insertion mutation)随机选择一个基因并将其插入染色体中一个随机的位置。位移变异随机选择子串，然后将其插入一个随机的位置。插入变异可以看作位移变异的特例，此时子串仅包含一个基因。*互反变异*(reciprocal exchange mutation)随机选择两个位置，然后交换这些位置上的基因。*移动变异*(shift mutation)随机选择一个基因，然后将其随机向左或向右移动到一个新的位置上。

6.2.4 以启发式方法为特点的遗传算子

受到一些成功的启发式方法的影响，针对作业车间调度问题创建了许多以启发式为特点的遗传算子。这类遗传算子的核心过程是启发式方法。如果一个方法可以将两个父

代合并产生两个子代，该方法就可以简单地看作杂交算子；如果可以改变某一父代特定数量的基因以产生一个子代，该方法就可以简单地看作变异算子。

基于 Giffler 和 Thompson 算法的杂交(algorithm-based crossover) Yamada 和 Nakano 提出了一种基于 Giffler 和 Thompson 算法的杂交算子[676]。Giffler-Thompson 算法的产生过程是树搜索方法。该算法的每一步要确定所有的加工冲突(对同一机器进行竞争的工序)，然后采用枚举过程来解决这些冲突。Yamada 和 Nakano 的杂交算子从本质上讲是一次通过过程(one-pass procedure)，而不是树搜索过程。在产生后代时，每步确定所有的加工冲突(正如 Giffler-Thompson 方法那样)，然后根据它们父代之一的调度从冲突的调度集合中选出一个。设：

o_{ji}——作业 j 的第 i 个工序；

S——给定的部分调度中可调度的工序集合；

φ_{ji}——S 中作业 j 的工序 i 的最早完成时间；

G_r——S 中机器 r 上冲突的工序集合。

由两个父代产生子代的过程如下：

基于 Giffler 和 Thompson 算法的杂交过程

第1步：初始假设 S 包含所有那些没有紧前工序(predecessor)的所有工序。

第2步：确定 $\varphi^* = \min\{\varphi_{ji} \mid o_{ji} \in S\}$ 和机器 r^*(φ^* 可以在该机器上实现)。

第3步：设 G_{r^*} 包括所有要求在机器 r^* 上加工的工序 $o_{ji} \in S$。

第4步：按照下面的方法从 G_{r^*} 中选择一个工序：

(4.1) 产生一个随机数 $\epsilon \in [0,1)$，然后将其与变异概率 p_m 进行比较。如果 $\epsilon < p_m$，则从 G_{r^*} 任意选择一个工序，记做 o_{ji}^*。

(4.2) 否则，按照相同的概率选择两个父代中的一个，称作 p_s，在 p_s 寻找最早出现在 G_{r^*} 中的一个工序，记做 o_{ji}^*。

(4.3) 根据 φ_{ji} 在后代中调度 o_{ji}^*。

第5步：按照下面的方法更新 S：

(5.1) 从 S 中删除 o_{ji}^*。

(5.2) 在 S 中添加 o_{ji}^* 的一个直接后续工序(direct successor)。

第6步：返回第2步直到产生一个完整的调度为止。

在第(4.1)步，通过随机选择一个工序来解决冲突，而在第(4.2)步，则通过优先选择父代 p_s 在 G_{r^*} 的所有冲突工序中最早出现的工序来解决冲突。父代 p_s 按照相同的概率随机选取。因此 Yamada-Nakano 方法本质上是基于优先分配启发式方法的，而不是纯粹的 Giffler-Thompson 方法。在每一步中选出一个工序添加到后代的部分调度，工序中的冲突通过根据其父代调度指定工序的优先级来解决。

基于邻域搜索的变异(Neighborhood Search-based Mutation) 在传统遗传算法中，

变异是用来在染色体上产生小扰动以维持种群多样性的次要算子。Cheng，Gen 和 Tsujimura 受邻域搜索技术的影响，提出了一种变异算子[115]。该算子不再是次要算子，用来执行深入的搜索以寻找改进的后代。

许多定义可以考虑作为一个调度的邻域。对基于工序的表示，给定染色体的邻域可以考虑为从给定的一个染色体通过交换 λ 个随机选择不重复的基因产生的染色体（调度）集合。如果一个染色体在某种度量方式下比其邻域中所有染色体都要好，则该染色体称作 λ 最优（λ-optimum）。

基于邻域搜索的变异过程

```
begin
    i←0;
    while(i≤pop_size×p_m)do
        随机选出一个未变异的染色体；
        从该染色体中随机选出 λ 个不相同的基因；
        利用这些基因所有的排列产生邻域；
        评价所有的邻域调度；
        选择最好的邻域染色体作为子代；
        i←i+1;
    end
end
```

Tsujimura 和 Gen 使用了信息熵作为种群中染色体多样性的度量[628]。这种基于熵的遗传算法既可以提供快速的收敛性能，还能够得到优质的解。

6.2.5 混合遗传算法

遗传算法环境中局部搜索的作用受到人们普遍的关注，许多成功的应用都得益于这样的混合方法。混合遗传算法常见的一种形式将局部搜索方法插入到遗传算法主循环中重组和选择部分之间。在这种混合算法中，遗传算法主要进行种群中的全局搜索，而局部搜索则进行染色体间的局部搜索。由于遗传算法和传统启发式方法的互补特性，混合式方法的效果通常要比任意一种都要好。混合的方式可以有许多种，如下所示：

1. 将启发式方法结合初始化以产生性能良好的初始种群。采用这种方法以后，带有最优性方法的混合遗传算法可以保证不比传统启发式方法的结果差。

2. 将启发式方法结合到评价函数中，用来将染色体解码为调度。

3. 将局部搜索启发式方法作为遗传算法基本循环的插件，让其与变异算子和杂交算子一同工作，以执行快速的局部搜寻，从而在对后代进行评价之前对其进行改进。

将遗传算法与局部搜索相结合 混合遗传算法的常见形式是将局部搜索与遗传算法相结合。遗传算法善长全局搜索但收敛速度慢；局部搜索善长微调但常会陷入局部最

优。混合方法结合了遗传算法和局部搜索启发式方法的特性。遗传算法用来执行全局搜索以使解从局部最优中逃离；局部搜索则用来进行性能微调。

这种情况下的局部搜索可以看作是对一个个体串生命周期学习过程的类比。在标准遗传算法中，染色体的选择是基于个体出生时的适应值，而在混合遗传算法中，选择则基于个体生命周期最终时的适应值，个体的生命由局部搜索来决定。得到改进的子代将局部搜索过程中学习到的特性通过常用的杂交算子传递给未来的子代[353]，这种现象称作拉马克进化(Lamarkian evolution)。于是这种混合方法可以看作是达尔文和拉马克进化的混合[194,468]。

遗传算法与局部搜索相结合的混合过程

```
begin
    t←0;
    初始化 P(t)(作业的顺序);
    执行局部搜索以改进染色体;
    评价 P(t);
    while(不满足终止条件) do
    begin
        重组 P(t);
        执行局部搜索以改进染色体;
        评价 P(t);
        选出下一代种群 P(t);
        t←t+1;
    end
end
```

将遗传算法与 Giffler 和 Thompson 的方法相结合 Dorndorf 和 Pesch 针对作业车间调度问题提出了一种基于优先规则编码的方法并开发了一种遗传算法的混合形式。该算法将著名的 Giffler-Thompson 算法[165]结合入遗传算法。在混合方法中，遗传算法用来进化一系列的优先分配规则，Giffler-Thompson 算法用来从优先分配规则编码中产生一个调度。混合方法的整体过程如下：

遗传算法与 Giffler 和 Thompson 算法相结合的混合算法过程

```
begin
    t←0;
    初始化 P(t)(作业的顺序);
    采用 Giffler Thompson 算法来产生调度;
    评价 P(t);
    while(不满足终止条件)do
    begin
```

```
            重组 P(t);
            采用 Giffler-Thompson 算法来产生调度;
            评价 P(t);
            选出下一代种群 P(t);
            t←t+1;
        end
    end
```

优先规则可能是实际应用中解决调度问题最常用的启发式方法，原因在于该方法易于实现，同时时间复杂度小。Giffler 和 Thompson 算法可以看作所有基于优先规则启发式方法的公共基础[590]。Giffler 和 Thompson 算法的产生过程是树结构的方法。树中的节点对应着部分调度，边表示可能的选择，树的叶则是枚举的调度的集合。对应一个给定的部分调度，该算法本质上确定所有的加工冲突（即对同一机器进行竞争的工序），然后采用枚举过程用各种方式在每一步骤中解决这些冲突[37]。Dorndorf 和 Pesch 采用了优先分配规则来替代枚举树搜索用以解决这些冲突，即一个基因指定了一个优先规则，该规则用于在一个时刻所有冲突工序中选出一个工序[165]。因此他们实际上采用了一次通过优先分配启发式方法，而不是纯粹的 Giffler-Thompson 算法。

产生过程在每一步骤中对一系列可调度的工序进行操作。可调度的工序即那些尚未被调度同时其紧前工序刚刚被调度的工序集合。该集合可以简单地通过前后结构获得。一次通过过程中步骤的数量与工序的数量相同，即 $m\times n$。每一步骤中，选出一个工序添加到部分调度(partial schedule)中。工序间的冲突由对应基因指定的优先分配规则来解决。与 Baker[37]书中的符号保持一致，我们有：

PS_t：包含 t 个已调度工序的部分调度

S_t：对应一个给定的 PS_t，步骤 t 中可调度工序的集合，

σ_i：所有 $i\in S_t$ 的工序可以开始的最早时间

ϕ_i：所有 $i\in S_t$ 的工序可以完成的最早时间

对应一个给定的有效部分调度，可能的开始时间 σ_i 由工序 i 的直接紧前工序的完成时间和工序 i 在机器上要求的最后完成时间确定。将这两个数中大的作为 σ_i。可能的完成时间 ϕ_i 就是 σ_i+t_i，其中 t_i 是工序 i 的加工时间。产生一个活动调度的过程如下所示：

基于优先规则编码(priority rule-based encoding)产生子代的过程

第 0 步：设$[p_1,p_2,\cdots,p_{mn}]$为一个给定的染色体。

第 1 步：设 $t=1$，起始 PS_t 设为空的部分调度，设 S_t 包含了所有没有紧前工序的调度。

第 2 步：确定 $\phi_t^*=\min\limits_{i\in S_t}\{\phi_i\}$，可实现 ϕ_t^* 的机器称为 m^*。如果存在多于一个这样的机器，通过随机选择来打破平局。

第 3 步：构造一个包含所有要在机器 m^* 上加工并且 $\sigma_i<\phi_t^*$ 的工序 $i\in S_t$ 的冲突集合 C_t。

根据优先规则 p_t 从 C_t 中选择一个工序，并将该工序添加到 PS_t 中尽可能早的位置，于是产生了一个新的部分调度 PS_{t+1}。如果根据优先规则 p_t 有多于一个工序满足要求，则通过随机选择来打破平局。

第4步：将被选择的工序从 S_t 中删除，并加入其直接后续工序。t 增加1。

第5步：返回第2步直到产生完整的调度为止。

余下的问题是如何确定有效的优先规则。希望阅读深入总结和讨论的读者可以参考 Panwalkar 和 Iskander[501]，Haupt[290]，以及 Blackstone，Phillips 和 Hogg[67] 的文章。表6.1包含了实践中经常使用的优先规则。

表6.1　作业车间分配规则

规　则	描　述
SPT(最短加工时间)	选择具有最短加工时间的工序
LPT(最长加工时间)	选择具有最长加工时间的工序
MWR(剩余最多工作)	选择所在作业具有最长剩余加工时间的工序
LWR(剩余最少工作)	选择所在作业具有最短剩余加工时间的工序
MOR(剩余最多工序)	选择所在作业具有最多剩余工序数的工序
LOR(剩余最少工序)	选择所在作业具有最少剩余工序数的工序
EDD(最早到期时间)	选择最早到期时间的作业
FCFS(先来先服务)	选择同一机器作业队列中最早的工序
RANDOM(随机)	随机选择一个工序

Kobayashi，Ono 和 Yamamura 提出了遗传算法与 Giffler-Thompson 算法另一种形式的结合方法[371,493]。他们的研究中采用了作业顺序矩阵编码方法。Giffler-Thompson 算法用来从编码中产生一个活动调度(active schedule)。产生活动调度的过程如下：

作业顺序矩阵编码(job-sequence matrix encoding)产生后代的过程

第1步：设 $t=1$，S_t 包含了所有没有紧前工序的调度。

第2步：确定 $\phi_t^* = \min_{i\in S_t}\{\phi_i\}$，可实现 ϕ_t^* 的机器称为 m^*。如果存在多于一个这样的机器，通过随机选择来打破平局。

第3步：构造一个包含所有要在机器 m^* 上加工并且 $\sigma_i<\phi_t^*$ 的工序 $i\in S_t$ 的冲突集合 C_t。

第4步：检查机器 m^* 上第一个未调度的工序。如果它属于冲突集 C_t，保持不变；否则将其与一个 C_t 中随机选择的工序交换。

第5步：将被选择的工序从 S_t 中删除，并加入其直接后续工序。t 增加1。

第6步：返回第2步直到所有工序均被调度为止。

将遗传算法与移动瓶颈启发式方法相结合　Dorndorf 和 Pesch 针对作业车间问题提出了基于机器编码，并开发了一种混合的遗传算法。该方法将著名的移动瓶颈启发式

方法结合进来[165]。在这种混合方法中，遗传算法用来进化机器顺序，而移动瓶颈启发式方法则用于从机器顺序编码中产生一个调度。

移动瓶颈启发式方法(shifting bottleneck heuristic)由 Adams，Balas 和 Zawack 提出[4]，可能是作业车间调度问题最强大的启发式方法。该方法将机器一个接一个排成顺序，每次从未被排序的机器中选出被确定为瓶颈的那个机器。每次对一个新的机器进行排序后，所有以前建立的顺利都要局部重新优化。瓶颈确定和局部重新优化过程都基于重复求解一个特定的单机调度问题，该问题是原始问题的松弛问题。该方法的主要贡献在于采用了这种松弛方法来决定对机器进行排序的次序。

移动瓶颈启发式方法基于给瓶颈机器优先级的传统思想。对机器的瓶颈量的不同度量将产生不同瓶颈机器的顺序。通过移动瓶颈启发式方法获得的调度质量深受瓶颈机器顺序的影响。Adams，Balas 和 Zawack 也提出了一种移动瓶颈启发式方法的枚举实现，这种实现考虑了不同的机器顺序。

Dorndorf 和 Pesch 采用了遗传策略(而不是枚举的树搜索方法)来确定移动瓶颈启发式方法中的最优机器顺序。一条染色体是排序的机器的列表。这里遗传算法用来进化这些染色体从而为移动瓶颈启发式方法找到更好的机器顺序。移动瓶颈启发式和遗传算法的不同之处在于在选择下一个机器时瓶颈不再是决策判据，而是由给定的染色体来确定，而且没有必要在每次迭代中确定瓶颈机器。

设 M_0 时已经排序的机器集合，给定的染色体是$[m_1, m_2, \cdots, m_m]$。从染色体中产生一个调度的过程如下：

基于机器编码(machine-based encoding)产生调度的过程

第 1 步：设 $M \leftarrow \varnothing, i \leftarrow 1$，染色体为$[m_1, m_2, \cdots, m_m]$。

第 2 步：对机器 m_i 进行最优排序。更新集合 $M_0 \leftarrow M_0 \cup \{m_i\}$。

第 3 步：对每个边界机器 $m_i \in M_0$ 依次重新优化顺序，保持其余顺序不变。

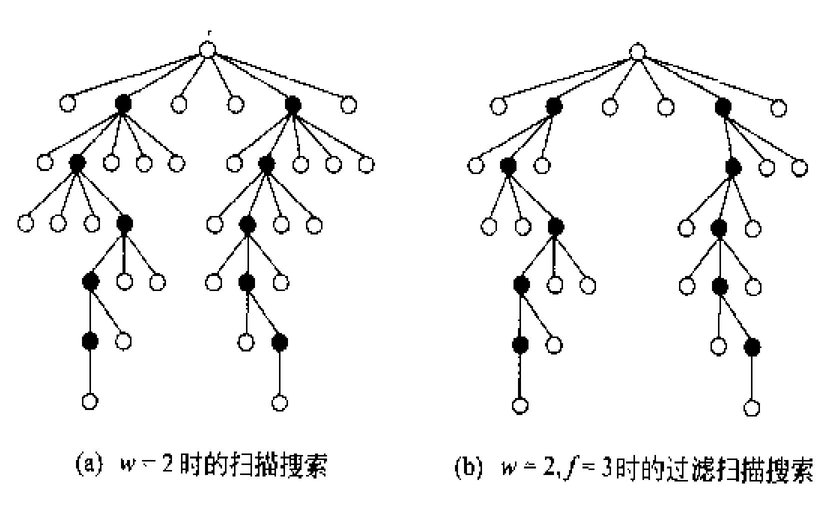
第 4 步：$i \leftarrow i+1$。如果 $i > m$，终止；否则返回第 2 步。

第 3 步的详细步骤可以参见 Adams，Balas 和 Zawack 的论述[4]或 Applegate 和 Cook 的论述[17]。

将遗传算法与扫描搜索(beam search)相结合　Holsapple 等人提出的混合方法将遗传算法与扫描搜索技术相结合[304]。粗略地讲，扫描搜索用于将染色体(基于作业的表示)转换为一个调度。扫描搜索是对宽度优先搜索的改善，该方法依赖于扫描宽度(beam width)的概念。这个概念限制了在搜索树的每个阶段(或层次)进行分支的节点数量。搜索过程的每一层次中，所有的节点都采用事先指定的评价函数进行评价。然后采用扫描宽度 w 来选择 w 个最佳的节点(根据他们的评价)在当前层次进行分支(即扩展)，目的是在树的下一层次中产生后代节点。当前层次中余下的节点被永久地剪除(prune)。如果仅有少于 w 的节点进行处理，则所有的节点都用来进行扩展。对于选出的 w 个节点中的

每个节点，产生该节点所有可能的后续节点。该过程一直持续到不可能产生进一步扩展为止。图6.1(a)用简单的例子说明了扫描搜索。

过滤扫描搜索是扫描搜索方法的改进。它采用另一个称作过滤宽度(filter width)的结构来进一步剪除状态空间。对于任意层次上选出用于扩展的节点的过滤宽度确定了其可以产生的后续节点的最大数量。即我们可以产生的后续节点的数量不能多于f。对于f的给定值，从可能的后续节点集合中随机选择出f个节点。w和f都是用户指定的数量。搜索过程如图6.1(b)所示。



(a) $w=2$时的扫描搜索　　(b) $w=2, f=3$时的过滤扫描搜索

图6.1　扫描搜索和过滤扫描搜索

Holsapple等人采用了与作业相关的表示(job-related representation)。遗传算法用来操作作业顺序，过滤扫描搜索用于针对给定的作业顺序(或一个染色体)产生"最优"调度。可以将扫描搜索看作评价函数的一部分，即将作业顺序(染色体的基因型)转换为一个调度(染色体的表现型)，当然扫描搜索的作用不止如此。Holsapple等人在柔性制造环境中考虑了一类静态的调度问题。该问题与典型作业车间问题的区别在于任意作业的工序都可以在任意机器上完成。在这种情况下，基于作业表示的染色体代表了许多可行的调度，于是他们采用了扫描搜索在给定染色体的可行调度中寻找"最优"调度。在典型作业车间调度问题中，加工每个工序的机器是事先指定的，因此基于作业的染色体仅代表一个可行的调度，正如前面所讨论的那样。在这种情况下不再需要诸如扫描搜索等树搜索方法将染色体转换为可行调度。Holsapple等人提出算法的整体过程如下：

遗传算法与扫描搜索的混合过程

begin

　　$t \leftarrow 0$;

　　初始化$P(t)$(作业的顺序)；

　　为每条染色体执行扫描搜索产生最优调度；

　　评价每个调度；

```
    while（不满足终止条件）do
    begin
        重组 P(t)；
        为每条染色体执行扫描搜索产生最优调度；
        评价每个调度；
        选出下一代种群 P(t)；
        t←t+1；
    end
end
```

6.2.6 讨论

在过去的10年里，作业车间调度问题成为遗传算法领域里的一个热门话题。原因是该问题表现出约束组合优化问题的所有特征，并且成为测试新算法思想的范例。注意遗传算法的一个解不一定是给定问题的一个解，但它可能包含了构造给定问题解的必要信息。正如我们所知，作业车间调度问题中有两种次序关系：(1)每台机器上的工序顺序；(2)每个作业工序的先后约束。第一个次序关系应由解方法确定；而第二个次序关系则应在调度中维持。如果染色体中混合了两种类型的次序关系，将使遗传算子的压力增大，需要同时处理不可行性和非法性问题。如何在遗传算法中表示作业车间调度问题的一个解是一个经过许多研究的问题。由于工序间存在先后约束，工序的排列通常对应着不可行解。许多研究人员提出的方法是用特定的调度生成器来对付麻烦的先后约束。这样作业车间调度问题就可以看作排列问题。遗传算法用于进化每台机器上的工序排列，调度生成器则用于根据排列和先后要求创建可行的解。Tsujimura 等人对三种编码方式：基于次序的表示、随机键表示和基于工序的表示对开放车间调度问题（一种作业车间调度问题的特例）进行了比较研究[630]。

大多数作业车间调度问题的编码方法都是一般字母串编码类型，即一个符号可以在染色体中重复出现。在这类方法中，包含若干子串的基于偏好列表的编码得到最广泛的应用。由于这种编码的每个子串是纯字母串，许多研究提出的方法就是将遗传算子应用于每个子串。这样，一般字母串可以从本质上作为若干纯字母串进行处理，于是许多针对纯字母串开发的遗传算子可以直接应用。这种遗传算子的多次使用会随着问题规模的增加而显著增加计算的时间。于是希望遗传算法的搜索能力也可以相应增加。值得注意的是，由于多个遗传算子在每个子串中独立应用，这样多次使用的后果之一就是可能违背工序间的先后约束。这就是需要生成器来重新调整工序顺序以获得可行解的原因。在编码复杂性和生成器之间的折衷需要进行考虑。

遗传作业车间调度实践中的一种潮流就是将局部搜索技术结合到遗传算法的主循环中，用来将每个子代转换为*活动调度*(active schedule)。一般来说，工序的可行排列对应

着半活动调度集合。随着问题规模的增加,半可行调度的规模会变得越来越大。由于遗传算法不善于微调,因此在半活动调度巨大的空间进行搜索将降低遗传算法的效率。已知作业车间调度问题的最优解位于活动调度集合中,该集合比半活动调度集合小得多。通过使用排列编码,无法将遗传搜索限制在活动调度中。一种可能的方法是让遗传搜索在整个半活动调度的搜索空间中进行,与此同时将每条染色体用附加的过程(调度生成器)转换为活动调度。这样做的一种方式是将 Giffler-Thompson 算法结合进遗传算法以执行转换。Giffler-Thompson 算法最初是一种枚举方法。在大多数这样的混合方法中,该算子作为一次通过启发式方法来重新调整工序的次序,解决先后约束,并将每个子代转换为活动调度。

最近 Bierwirth 和 Mattfeld 提出了一种遗传算法[709]。该算法用于解决描述静态、动态和非确定性生产环境的一般作业车间调度问题。

6.3 群体作业调度问题

自从成组技术(group technology,GT)成为工业生产系统中广泛使用的技术以来,群体作业的生产调度就成为活跃的研究领域[657]。在 GT 中,作业根据其相似性被划分为若干类。一个作业在跟随同一类另一个作业时不需要进行设置,但当其跟随其他类的作业时就需要进行类设置。**群体作业调度问题**(grouped job scheduling problem)的计算复杂性在类何时被分离的问题上是不确定的。这意味着如果没有在调度中需要刚好 m 个设置的简单假设,计算可能非常复杂[461,514]。

群体作业调度问题从工业实践中产生。举例来说,群体部分在一个机器工具机件中进行加工。不同尺寸的钢钉被一个钢铁架的不同滚筒碾磨,多个产品在化工厂的批量生产模式中生产。二者的生产调度都可以描述为群体作业调度问题[15,459]。群体作业调度问题的求解方法主要包括动态规划和启发式方法[15,518]。Wang,Cheng 和 Gen 开发了一种遗传算法来求解该问题[651]。

6.3.1 问题的描述和必要条件

为调度群体作业而最小化单个机器的总流程时间问题可以描述如下:假设有 N 个作业在一个机器中心等候处理。这些作业可以根据其相似性划分为 m 个群体。群体 i 有 n_i 个作业。群体 i 中第 j 个作业表示为对(i,j)。作业(i,j)的加工时间为 p_{ij}。群体 i 的类设置时间(family setup time)为 $s_i>0,i=1,2,\cdots,m$。一个调度包含一系列的运行(run)。一个运行就是两类设置间加工的作业顺序[657]。对于一个给定的调度,设 η 表示运行的数量。将第 i 次运行记做 R_i。设 β_i 表示 R_i 中作业的数量,$[i]$表示 R_i 中加工的类的下标,$[i,j]$表示 R_i 中加工的第 j 个作业的作业对。定义调度中 R_i 的位置 K_i 为

$$K_i = \sum_{k=1}^{i-1} \beta_k + 1, \quad i = 1, 2, \cdots, \eta \tag{6.1}$$

问题就是寻找最优调度 S 来最小化总流程时间 $F(S)$：

$$\min F(S) = \sum_{i=1}^{\eta} \left[(N - K_i + 1) S_{[i]} + \sum_{j=1}^{\beta_i} (N - K_i - j + 2) p_{[i,j]} \right] \tag{6.2}$$

对于运行 i，定义其机器占用时间为 q_i，R_i 中作业的平均机器占用时间为 v_i。有：

$$q_i = s_{[i]} + \sum_{j=1}^{\beta_i} p_{[i,j]}, \quad i = 1, 2, \cdots, \eta \tag{6.3}$$

$$v_i = q_i / \beta_i, \qquad i = 1, 2, \cdots, \eta \tag{6.4}$$

问题等效于最小化总的加权流程时间(所有的权重系数都是 1)。从 Webster 和 Baker 文献[657]中的性质 2 出发，Wang 证明了群体作业调度问题最优解的必要条件[648]：

性质 6.1 在一台机器上最小化群体作业的总流程时间，最优解满足下面的条件：

1. 如果 $p_{ij} \leqslant p_{ik}$，作业(i,j)在作业(i,k)前加工。
2. 运行采用 SMMOT(最短平均机器占用时间)次序，即

$$v_1 \leqslant v_2 \leqslant \cdots \leqslant v_\eta \tag{6.5}$$

3. 如果$[i]=[k]$，而且 $i<k$，则

$$p_{i\beta_i} \leqslant v_{i+1} \leqslant \cdots \leqslant v_{k-1} \leqslant p_{k1} \tag{6.6}$$

4. 如果

$$\frac{s_i + \sum_{k=1}^{j} p_{ik}}{j} \geqslant p_{ij} \tag{6.7}$$

则从作业$(i,1)$到作业$(i,j+1)$进行顺序加工。

很明显性质 6.1 给出了最优解的必要条件。为了讨论如何维持必要条件，给出了下面的定义：

定义 6.1 满足性质 6.1 的调度称作合理调度(reasonable schedule)。合理调度具有如下性质。

性质 6.2 在合理调度中，R_i 和 R_j 是同一类中的运行，R_i 在 R_j 前面。如果 R_k 通过 R_i 和 R_j 的组合产生，其一个作业的平均占用机器时间 v_k 满足

$$v_k \leqslant v_j \tag{6.8}$$

证明 由于 $v_i \leqslant v_j$，

$$\begin{aligned} v_k &= \frac{1}{\beta_i + \beta_j} (\beta_i v_i + \beta_j v_j - s_{[j]}) \\ &\leqslant \frac{1}{\beta_i + \beta_j} (\beta_i v_j + \beta_j v_j - s_{[j]}) \end{aligned}$$

$$= v_j - \frac{s_{[j]}}{\beta_i + \beta_j}$$

鉴于事实 $s_{[j]} \geqslant 0$，我们有 $v_k \leqslant v_j$。

性质 6.2 告诉我们合理调度同一类中两次运行的组合将在最后一次运行之前加工。

性质 6.3 如果同一类中两次运行的组合移动到调度中的第 k 个位置，维持其合理的条件就是同一类中没有其他运行在位置 i,k 和 k,j 之间。

从性质 6.3 显然可以知道合理调度中运行的组合和移动无法越过同一类中的任何运行，原因是这会使性质 6.1 中的必要条件 1 无法满足。

6.3.2 基本运行

基本运行的定义基于必要条件和合理调度的性质。产生基本运行的过程描述如下：

基本运行过程

第 1 步：设 $\eta = N$，将所有相同的作业看作独立运行。按照 $s_i + p_{ij}$ 增加的顺序对这些运行进行排序。显然，这是初始的合理调度。有 $v_1 \leqslant v_2 \leqslant \cdots \leqslant v_\eta$，其中 $v_i = s_{[i]} + p_{[i,1]}$。

第 2 步：设 $\eta' = \eta$。从 $k=1$ 到 η，检查：如果 R_{k+1} 是 R_k 同一类中的一个运行，将 R_{k+1} 与 R_k 进行组合，设 $\eta = \eta - 1$。

第 3 步：如果 $\eta = \eta'$，终止；否则更新 $v_i, i = 1,2,\cdots,\eta$。

第 4 步：对于 $k=1$ 到 η，检查：如果 $v_{k+1} < v_k$，将 R_{k+1} 前移直到 $v_{k+1} \geqslant v_i$ 或 $[i]=[k+1]$ 为止（即 R_i 和 R_{k+1} 是同一类中的运行）。将 R_{k+1} 定位在 R_i 的紧后方，将所有的运行重新标记顺序。返回第 2 步。

定义 6.2 定义由基本运行算法产生的调度为*基本调度*（fundamental schedule），基本调度中的运行是基本运行。

容易看出基本调度可以满足性质 6.1 中最优解的必要条件。对于大多数实际问题，群体的数量通常远小于作业的数量。经验表明，当 $m \ll N$ 时基本运行的数量少于作业的数量。可以证明最优解可以通过基本运行的组合来获得。因此仅有必要考虑 η 个基本运行来替代求解过程中的 N 个作业。

引理 6.1 通过将运行 R_i 从 K_i 移动到 K 对总流程时间增加的贡献为 $(K_i - K)q_i$。

证明 将增加的贡献表示为 $\Delta F_i(K_i, K)$。由目标函数(6.2)，我们有：

$$\begin{aligned}
\Delta F_i(K_i,K) &= (N-K+1)s_{[i]} + \sum_{j=1}^{\beta_i}(N-K-j+2)p_{[i,j]} \\
&\quad - (N-K_i+1)s_{[i]} - \sum_{j=1}^{\beta_i}(N-K_i-j+2)p_{[i,j]} \\
&= (K_i - K)s_{[i]} + (K_i - K)\sum_{j=1}^{\beta_i} p_{[i,j]} \\
&= (K_i - K)q_i
\end{aligned}$$

定理 6.1 群体作业调度问题在单机上最小化总流程时间的最优解是基本运行的组合。

证明 假设最优调度不是基本运行的组合。于是存在一个基本运行，该运行分离为最优调度中至少两个部分。假设基本调度 S0 构造如下：

$$\text{S0:}\quad * + R_1 + R_0 + R_2 + *$$

其中 R_0 是一个基本运行，R_1 和 R_2 是基本运行的部分邻域，* 表示调度中的其他部分，+表示运行的邻域关系。

如果最优调度中插入 R_1，将 R_0 分离为两个子运行 R_{01} 和 R_{02}，即

$$\text{S1:}\quad * + R'_1 + R_{01} + R_1 + R_{02} + R_2 + *$$

其中 R'_1 是 R_1 为了构成基本运行的补充部分。

由性质 6.1 我们有

$$v(R'_1) \leqslant v(R_{01}) \leqslant v(R_1) \leqslant v(R_{02}) \leqslant v(R_2) \tag{6.9}$$

其中 $v(R)$ 代表一个作业在运行 R 中的平均机器占用时间。根据基本运行(FR)过程，在基本调度的组合过程中存在三种可能的情况：

$$\text{C1:}\quad * + R'_1 + R_1 + R_{01} + R_{02} + R_2 + *$$

$$\text{C2:}\quad * + R_{01} + R'_1 + R_1 + R_{02} + R_2 + *$$

$$\text{C3:}\quad * + R_{01} + R_{02} + R'_1 + R_1 + R_2 + *$$

对于情况 1，从(C1)得到 $v(R_1) \leqslant v(R_{01})$，将其与(6.9)比较有：

$$v(R_{01}) = v(R_1)$$

交换最优调度 S1 中 R_{01} 和 R_1 的位置，容易看出交换使得 R_{01} 反向移动 β_1 个作业而 R_1 正向移动 β_{01} 个作业。将新调度表示为 SE。由引理 6.1 我们有：

$$\begin{aligned} F(\text{SE}) - F(\text{S1}) &= \beta_{01} q_1 - \beta_1 q_{01} - s_0 K_{02} \\ &= (q_1/\beta_1 - q_{01}/\beta_{01})\beta_1\beta_{01} - s_0 K_{02} \\ &= (v(R_1) - v(R_{01}))\beta_1\beta_{01} - s_0 K_{02} \\ &= - s_0 K_{02} < 0 \end{aligned}$$

这与调度 S1 的最优性矛盾。

对于情况 2，由(C2)和式(6.9)可以知道 $v(R'_1) = v(R_{01})$。通过将 R'_1 与 R_{01} 交换可以类似地证明 S1 不是最优的。对于情况 3，由(C3)和式(6.9)可以知道 $v(R_1) = v(R_{02})$。通过将 R_1 与 R_{02} 交换也可以类似地证明 S1 不是最优的。如果 R_0 被 R_2 在最优调度中分离，我们也可以证明调度不是最优的。基于定理 6.1，可以使用 FR 过程产生基本运行。于是最优调度可以通过组合基本运行得到。我们可以由性质 6.1 中条件 4 来组成运行，但那并不好。即通过条件 4 产生的运行数量大大超过基本运行数量。

6.3.3 表示

Wang,Gen 和 Cheng 针对群体作业车间调度问题开发了一种遗传算法。该算法采用基本的二进制串来表示基本运行的组合。假设基本调度如下：

$$R_{(1)} + R_{(2)} + \cdots + R_{(\eta)}$$

在类 $i(i=1,2,\cdots,m)$ 中有 η_i 个基本运行。显然有

$$\eta = \sum_{i=1}^{m} \eta_i \tag{6.10}$$

定义 $\eta_0=0$ 以及

$$\bar{\eta}_i = \sum_{k=1}^{i} \eta_k, \quad i = 1,2,\cdots,m \tag{6.11}$$

于是这些运行可以被划分为类，同时标记如下：

类 1：$R_1, R_2, \cdots, R_{\bar{\eta}_1}$

类 2：$R_{\bar{\eta}_1+1}, R_{\bar{\eta}_1+2}, \cdots, R_{\bar{\eta}_2}$

⋮

类 m：$R_{\bar{\eta}_{m-1}+1}, R_{\bar{\eta}_{m-1}+2}, \cdots, R_{\eta}$

由于无法组合不同类中的运行，在类 k 中最后的运行和类 $k+1$ 中最先的运行的组合中没有位来表示组合($k=1,2,\cdots,m-1$)。于是一个有 $\eta-m$ 个位的二进制数就可以用作基本运行组合的基因表示。定义基因表示 L 的二进制串为：

$$L = \eta - m \tag{6.12}$$

因此如果运行 R_{i-1} 和 R_i 属于同一类 k，则定义运行之间组合的二进制位为

$$b_{i-k} = \begin{cases} 1, & R_{i-1} \text{ 和 } R_i \text{ 被组合} \\ 0, & \text{否则} \end{cases} \quad k = 1,2,\cdots,m \quad i \in [\bar{\eta}_{k-1}+2, \bar{\eta}_k] \tag{6.13}$$

b_{i-k} 和 R_{i-1}, R_i 之间的关系如下：

$$\underbrace{R_1 -}_{b_1} \underbrace{R_2 - R_3}_{b_2} - \cdots - \underbrace{R_{\bar{\eta}_1-1} - R_{\bar{\eta}_1}}_{b_{\bar{\eta}_1-1}} \mid \underbrace{R_{\bar{\eta}_1+1} - R_{\bar{\eta}_1+2}}_{b_{\bar{\eta}_1}} - \cdots - \underbrace{R_{\eta-1} - R_{\eta}}_{b_{\eta-m}}$$

于是

$$B = [b_1, b_2, \cdots, b_L] \tag{6.14}$$

是基本运行的组合表示。举个简单例子，基本调度为

$$R_1 + R_4 + R_2 + R_5 + R_3$$

归类后如下：

类 1：R_1, R_2, R_3

类 2：R_4, R_5

于是一个有 5－2＝3 位的二进制串可描述这些运行的组合。位和运行之间的关系为

$$\underbrace{R_1 -}_{b_1} \underbrace{R_2 - R_3}_{b_2} \underbrace{R_4 - R_5}_{b_3}$$

比如(1 0 1)表示 R_1 和 R_2 进行组合,R_4 和 R_5 也进行组合,但 R_2 和 R_3 不组合。

6.3.4 评价

设 $B(j), j=1,2,\cdots,pop_size$ 表示当前种群的个体。它们的适应值可以由下面的过程来计算:

适应值函数过程

第 1 步:对于个体 $j(j=1,2,\cdots,L)$将其对应的运行与等于 1 的位相组合,并对所有运行的组合计算作业的平均机器占用时间 $v(C_i)$,其中 C_i 代表第 i 个组合。

第 2 步:按照下面的方式对组合进行排序和重标记

$$v(C_1) \leqslant v(C_2) \leqslant \cdots \leqslant v(C_{NC}) \tag{6.15}$$

其中 NC 是个体 j 中组合的数量。容易看出 NC$=L-$($B(j)$中等于 1 的位的数量)。

第 3 步:根据式(6.2)计算 $B(j)$的目标函数 $F(j)$。

第 4 步:通过下式计算个体 j 的适应值函数

$$f(j) = F_{max} - F(j) \tag{6.16}$$

其中

$$F_{max} = \max\{F(j) \mid j = 1,2,\cdots,pop_size\} \tag{6.17}$$

从式(6.16)中容易看出调度质量越高适应值越大。

6.3.5 遗传算子

采用了单分割点杂交(one-cutpoint crossover)。在杂交操作中,使用一对父代,一个用轮盘比例方法选出,另一个则随机选取。设 $P(j)$是个体 j 的选择概率,定义为:

$$P_j = \frac{f(j)}{\sum_{k=1}^{pop_size} f(k)}, \quad j = 1,2,\cdots,pop_size \tag{6.18}$$

其中 $f(k)$由式(6.16)定义。显然适应值函数较大的个体被选择的概率也较大。为了选择一个后代 s ($s=1,2,\cdots,N$),定义 $S(j)=P(1)+P(2)+\cdots+P(j)$ ($j=1,2,\cdots,N$)并产生一个随机数 $\xi_s \in U(0,1)$(其中 $U(0,1)$是(0,1)区间上的均匀分布[68]。如果 $S(j-1)<\xi_s<S(j)$,将个体 j 选作后代 s 的一个父代[455]。

6.3.6 整体过程

整体过程描述如下:

遗传算法过程

第 1 步：指定最大遗传代数 max_gen，种群规模 pop_size、杂交概率 p_c 和变异概率 p_m。对于任意的 i,j 输入作业数据 s_i 和 p_{ij}。

第 2 步：产生基本调度 FS，并将所有的基本运行标记为一类顺序。由式(6.12)确定二进制串的长度 L。

第 3 步：产生初始种群。对于 $B(j)$ $(j=1,2,\cdots,pop_size)$ 通过下式产生位：

$$b_i=\begin{cases}1, & \xi_i>0.5\\ 0, & \text{否则}\end{cases}\qquad i=1,2,\cdots,L$$

其中 $\xi_i\in U(0,1)$，$U(0,1)$ 是 $(0,1)$ 上的均匀分布。

第 4 步：通过下式寻找最佳初始个体

$$j^*=\arg\min\{F(j)\mid j=1,2,\cdots,pop_size\}$$

设 $F^*=F(j^*)$，$B^*=B(j^*)$。设置初始迭代下标 $k=0$。

第 5 步：设置迭代下标 $k\leftarrow k+1$。如果 $k=max_gen$，输出结果 F^* 和 B^*，终止算法；否则继续迭代。

第 6 步：计算所有个体的目标和适应值函数。通过下式寻找最佳的个体：

$$j^*=\arg\min\{F(j)\mid j=1,2,\cdots,\mathrm{NP}\}\tag{6.19}$$

如果 $F(j^*)<F^*$，则 $F^*=F(j^*)$，$B^*=B(j^*)$。

第 7 步：由式(6.18)计算所有个体的选择概率。计算

$$S(j)=P(1)+P(2)+\cdots+P(j),\quad j=1,2,\cdots,N\tag{6.20}$$

第 8 步：对于 $s=1,3,5,\cdots,pop_size$，通过复制产生子代(新个体)。产生一个随机数 $\xi_s\in U(0,1)$。如果 $S(j-1)<\xi_s<S(j)$，将个体 j 选作一个父代并将 $k=\mathrm{int}[\mathrm{NP}\cdot\xi_{s+1}]+1$ 作为子代 s 和 $s+1$ 的另一个父代。根据概率 p_c 和 p_m 进行杂交和变异。

第 9 步：更新 $B^k(j)\leftarrow B^{k+1}(j)$，$j=1,2,\cdots,pop_size$，返回第 5 步。

6.3.7 数值例子

考虑一个磨粉工厂接受 20 个订单的小型测试问题。每个订单(作业)需要被 5 种不同类型滚筒的其中一种进行碾压。订单的碾压时间和滚筒的设置时间见表 6.2。根据基本运行算法，将所有相同的作业看作独立运行并将其按照 S_i+p_{ij} 增加的顺序进行排列，即

$$11+6+\underbrace{1+2+3}+10+12+5+\underbrace{8+15}+4+9+20+7$$

$$+\underbrace{17+14+16+19}+13+18$$

表 6.2 订单的碾压和设置时间

订单	碾压时间	滚筒	设置时间	订单	碾压时间	滚筒	设置时间
1	8			11	12		
2	10	1	3.0	12	8		
3	15			13	19	4	1.0
4	25			14	95		
5	18	2	7.0	15	19		
6	4			16	21		
7	19			17	8	5	31.0
8	12			18	75		
9	17	3	31.0	19	23		
10	5			20	1		

目标值为 4876。将相邻作业组合为同一群体((1+2+3),(8+15),(17+14+16+19)),然后按照 v_i 增加的顺序对运行进行排列。重复这个过程。最终我们得到包含了 10 个基本运行的基本调度:

$$R_6+R_2+R_1+R_4+R_7+R_9+R_3+R_5+R_8+R_{10}$$

将其分解如下:

1. 类 1 R_1:作业 1、2、3
2. 类 2 R_2:作业 6;R_3:作业 5、4
3. 类 3 R_4:作业 10、8、15;R_5:作业 9、7
4. 类 4 R_6:作业 11;R_7:作业 12;R_8:作业 13
5. 类 5 R_9:作业 20、17、14、16、19;R_{10}:作业 18

基本调度的目标值为 3551。

根据式(6.12),二进制串的长度为 10−5=5。因此基因表示为

$$R_1 \mid \underbrace{R_2 \quad R_3}_{b_1} \mid \underbrace{R_4-R_5}_{b_2} \mid \underbrace{R_6-R_7}_{b_3}\underbrace{-R_8}_{b_4} \mid \underbrace{R_9-R_{10}}_{b_5}$$

设 $pop_size=10$,$max_gen=50$。通过运行算法得到的最佳调度为

$$B^*=11001, \quad F^*=3393$$

最佳调度为

$$R_6+R_1+\underbrace{R_4+R_5}+\underbrace{R_2+R_3}+R_7+\underbrace{R_9+R_{10}}+R_8$$

或

$$11 \mid +1+2+3 \mid +10+8+15+9+7 \mid +6+5$$
$$+4 \mid +12 \mid +20+17+14+16+19+18 \mid +13$$

这个解通过分支定界算法验证为最优解。该解通过组合基本运行 R_2+R_3,R_4+R_5 和 R_9+R_{10} 得到。

6.4 资源约束的项目调度

在资源和前后约束下带有最小化项目持续时间目标的活动(activity)调度问题在文献中被称作资源约束的项目调度(resource-constrained project scheduling)问题[37]。该问题的基本类型可以描述如下：一个项目包含许多相互关联的活动，其中每一个都有确定的持续时间和给定的资源要求。资源的数量是有限的，但会随着时间而更新。资源之间不可替代，活动不能被中断。问题的解是确定考虑前后约束和资源约束的活动开始时间以优化目标。

作为例子，考虑一个简单的包含5个活动的项目(如图6.2所示)。节点表示活动，有向连线表示先后约束。假设总的资源数量为4个单位。每个活动的持续时间和资源消耗见表6.3。

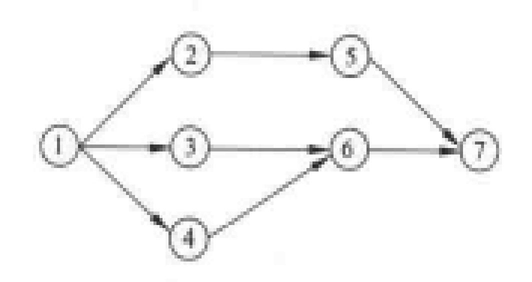


图6.2 一个项目的网络表示

表6.3 每个活动的持续时间和资源消耗

活动	持续时间	资源消耗	父活动
1	虚拟活动		
2	4	2	1
3	3	3	1
4	2	2	1
5	4	1	2
6	2	2	3,4
7	虚拟活动		

如果项目的调度如图6.3所示，则所需资源随着时间的变化(即资源数据图)用图6.4表示。

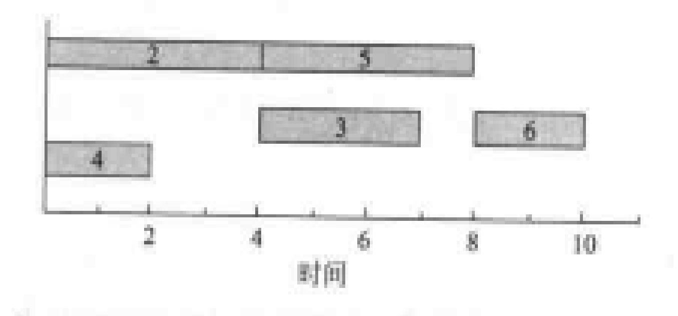


图6.3 Gantt图调度

该问题可以用数学语言表示为

$$\min \quad t_n \tag{6.21}$$

$$\text{s.t.} \quad t_j - t_i \geqslant d_i, \quad \forall j \in S_i \tag{6.22}$$

$$\sum_{i \in A_{t_i}} r_{ik} \leqslant b_k, \quad k = 1,2,\cdots,m \tag{6.23}$$

$$t_i \geqslant 0, \quad i = 1,2,\cdots,n \tag{6.24}$$

其中 t_i 是活动 i 的开始时间，d_i 是活动 i 的持续时间（进行时间），S_i 是在活动 i 之前的活动的集合，r_{ik} 是活动 i 需要资源 k 的数量，b_k 是资源 k 的总量，A_{t_i} 是 t_i 时刻进行的活动的集合，m 是不同资源类型的数量。活动 1 和 n 是虚拟活动，其作用是标记项目开始和结束的时间。目标是最小化总的项目持续时间。约束(6.22)确保不违背前后约束。约束(6.23)确保任何时段内所有活动消耗的资源 k 的数量不超过其限制的数量。该研究主要关注如何处理问题中的前后约束。有人提出了一种新的编码方法，该方法本质上可以对于给定的实例表示活动所有的可行排列[105,106,110]。

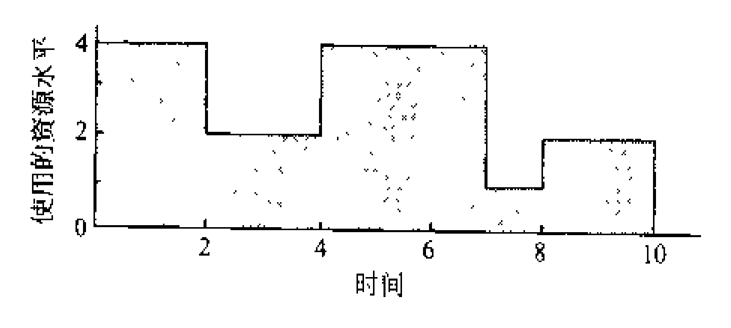


图 6.4 资源数据图

早期的尝试是使用数学规划中的标准求解方法来寻找问题的准确最优解[121,143,506,589,611]。由于资源约束的项目调度问题是 NP 难的，对于大的项目来说，问题的规模可能致使寻求最优解的方法在计算上不现实。在这种情况下，该问题应该采用启发式问题求解方法来求解，这些方法采用相当简单的调度规则来合理地产生次最优调度。在过去的 30 年里，提出并测试了大量启发式算法，我们推荐读者参考 Alvarez-Valdés 和 Tamarit 的综述与计算比较[12]。

现有的大多数启发式方法可以看作优先调度规则，它们在为解决资源冲突而进行顺序决策时根据临时或与资源相关的启发式规则来分配活动的优先权。最近 Cheng 和 Gen 提出了一种混合遗传算法用于求解资源约束的项目调度问题。该问题本质上包含以下两个基本部分：(1)确定不违反前后约束的活动进行次序；(2)然后确定不违反资源约束的每个活动的开始时间。如何确定活动的次序是问题的关键，原因在于一旦活动次序确定，可以容易地根据次序来构造一个调度。Cheng 和 Gen 方法的基本思想是(1)采用遗传算法来进化适当的活动进行次序；(2)采用最佳配合过程来计算活动的开

始时间。

6.4.1 基于优先权的编码

如何将问题的解编码为染色体是遗传算法其余步骤进行的先决重要问题。在 Cheng 和 Gen 方法中，采用遗传算法来进化适当的活动进行次序，因此仅须对活动次序进行编码。这本质上是排列问题。由于活动之间存在前后约束，任意的排列可能产生不可行的进行次序。如何有效地产生能够处理前后约束的编码是该方法的关键步骤，也是其余步骤的先决条件。他们提出了一种基于优先权的编码方法来处理这个困难，该方法来源于有向无环图模型。

有向无环图(directed acyclic graph) 一个项目的实例可以用有向无环图来表示[9]。一个有向无环图(directed acyclic graph) $G=(V,A)$包含一个代表活动的节点的集合 V 和一个代表活动间前后约束的有向边的集合 A。术语节点(node)和活动(activity)在以下儿小节中交替使用。图 6.2 给出了 DAG 表示一个项目的例子。对所有活动进行排序以满足前后约束的问题等效于产生一个下面定义的 DAG 的拓扑排序[134]。

定义 6.3(拓扑排序(topological sort)) 对于一个给定的有向图 $G=(V,A)$，一个拓扑排序是一个所有节点的线性次序，对于任意有向边$(u,v)\in A$，u 在该次序中先于 v 出现。

图 6.5 表示了图 6.2 例子的拓扑排序。节点的排列确保所有有向边的方向从左到右。

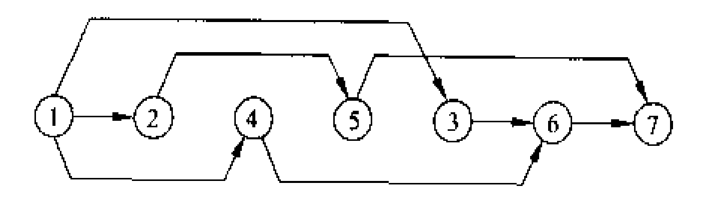


图 6.5 图 6.2DAG 的拓扑排序

容易看出活动的随机排列通常不能对应给定 DAG 的拓扑排序，因此排列编码(实际遗传算法最常用的编码方式)对该问题不适用。Cheng 和 Gen 提出了一种称作基于优先权编码(priority-based encoding)的新编码方式。该方法 (1)可以表示给定实例所有可能的拓扑排序；(2)编码的任何排列总是对应着一个拓扑排序，即一个可行解。

基于优先权的编码(priority-based encoding) 遗传算法的基因包含两种信息：基因座(locus)(基因在染色体结构中的位置)和等位基因(allele)(基因的值)。因此有两种可能的方式来表示一个活动：采用基因座和采用等位基因。这里用位置来表示一个活动的ID，基因的值用来表示活动的优先权，如图 6.6 所示。基因的值是一个$[1,n]$之间惟一的整数。数值越大则优先权越高。

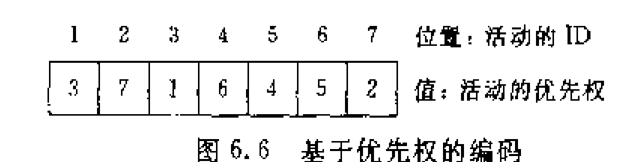


图 6.6 基于优先权的编码

采用了一次通过过程来由染色体产生拓扑排序，即一次性从左到右确定活动。当进行关于一个位置的决策时，若干活动可能就这一位置进行竞争，具有最高优先权的一个赢得该位置。这种编码不是直接表示给定 DAG 的拓扑排序。它仅包含了用于解决冲突的某些信息。大多数情况下可以根据编码来惟一地确定拓扑排序。优先权的任何改变通常导致不同的拓扑排序。因此这种编码本质上能够表示给定 DAG 所有可能的拓扑排序。

下面考察如何从编码产生一个拓扑排序。考虑图 6.2 表示的例子。向量 $\boldsymbol{A}[\cdot]$用来存储产生的拓扑排序。最初 $\boldsymbol{A}[1]=1$。然后三个活动 2,3,4 来竞争 $\boldsymbol{A}[2]$。它们的优先权分别是 7,1,6。由于活动 2 具有最高的优先权，它赢得该位置。在固定 $\boldsymbol{A}[2]=2$ 以后，下一个位置 $\boldsymbol{A}[3]$的竞争者是活动 3,4,5。活动 4 赢得该位置并固定 $\boldsymbol{A}[3]=4$。重复下面两步：

第 1 步：对于当前位置构造一个候选集合。

第 2 步：选择具有最高优先权的活动，直到活动一个图 6.5 所示的拓扑排序。

拓扑排序(topological sort) 产生一个拓扑排序的过程按照一次通过方式进行：从左到右产生一个拓扑排序，一次固定一个节点的次序。该过程的每一次迭代中所有的节点都处于下列三种状态之一：

1. 已排序节点(sorted node)：已经在构造的拓扑排序中的节点
2. 合格节点(eligible node)：那些所有父节点都是已排序节点的节点
3. 自由节点(free node)：所有其他节点

当然关键部分就是如何寻找合格节点的集合。下面的定义和定理解释了如何产生这样的集合以及过程如何进行。

对于有向图 $G=(V,A)$的边(u,v)，我们称该边从 u 传出(incident from u)以及传入 v(incident to v)。此外称 u 与 v 相临近，v 也与 u 相临近。节点 u 的内度(indegree)，用 $d^{\mathrm{IN}}(u)$表示，就是传入节点 u 的节点数量。图 G 的一个割集是一些边的集合，该集合具有$(X,\ V-X)$的形式，其中 $u\in X, v\in V-X$。通常用 $\overline{X}$ 来表示 $V-X$。一个部分拓扑排序(partial topological sort)就是一个仅包含固定次序的前 $t(t<|V|)$个节点的排序。设 $PS_t\subset V$ 表示关于给定部分拓扑排序的节点集合，其中下标 t 代表集合的规模，即$|PS_t|=t$。设 $C(PS_t,V-PS_t)=\{(i,j)|i\in PS_t, j\in V-PS_t\}$表示关于给定部分拓扑排序的有向图的割集。于是有下面的引理。

引理 6.2(合格节点) 对于一个给定的节点为 PS_t 的部分拓扑排序，节点 $j\in V-PS_t$

是合格的当且仅当存在传入 j 的边集合 $S(j)=\{(i,j)|(i,j)\in A\}$满足 $S(j)\subseteq C(PS_t,V-PS_t)$。

证明 对于给定的节点 $j\in V-PS_t$，如果存在一条边 (x,j) 传入 j，节点 x 是节点 j 的父节点。如果所有的这样的边都属于割集 $C(PS_t,V-PS_t)$，这意味着所有节点 j 的父节点都属于 PS_t，即它们是已排序节点，因此节点 j 是合格的。

假设对于合格节点 j，并非所有传入 j 的边都属于割集。即 $|C(PS_t,V-PS_t)\cup S(j)|>|C(PS_t,V-PS_t)|$。于是至少一个父节点属于集合 $V-PS_t$，即至少其一个父节点不是已排序节点。这与合格节点的定义矛盾。

理论上我们可以通过本引理来检查一个节点是否合格，但很难程序化地检查是否一个集合是另一个集合的子集。下面的定理提供了确定一个合格节点的判据。

定理 6.2(合格节点判据) 对于一个合格节点 $j\in V-PS_t$，设 $S_t(j)$是割集 $C(PS_t,V-PS_t)$的一个包含所有传入 j 的边的适当子集。于是我们有 $|S_t(j)|=d^{IN}(j)$。

证明 对于一个合格节点，根据引理 6.2 有 $S_t(j)\cap S(j)=S(j)$，而且 $S_t(j)\cup S(j)=S(j)$。由于 $|S(j)|=d^{IN}(j)$，于是证明了定理。

现在可以仅通过验证割集中传入一个节点的边的数量是否等于其内度来检查该点是否为合格节点。这个判据易于编程。让我们考虑图 6.7 给出的例子。部分拓扑排序是 $PS_3=\{1,2,3\}$，割集包括有向边 $C(PS_3,V-PS_3)=\{(1,4),(2,5),(2,6),(3,6),(3,7)\}$。由于节点 6 的内度 $d^{IN}(6)=2$ 而且传入该节点的两个边属于割集，因此节点 6 是合格节点。由于节点 7 的内度为 2，而仅有 1 条属于割集的边传入该节点，因此节点 7 是自由节点。该节点的一个父节点是节点 4，它是一个合格节点，不是已排序节点。

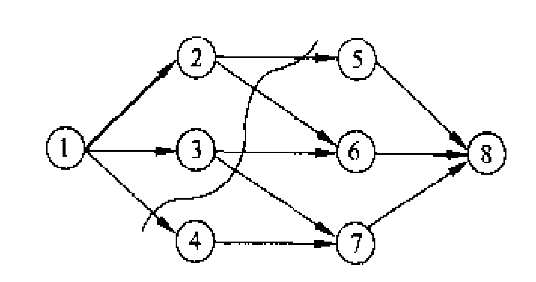


图 6.7 部分拓扑排序、割集和合格节点

拓扑排序过程的基本思想是，在过程进行的每一步都要 (1)根据定理 6.2 确定合格节点集合；(2)将该集合中具有最高优先级的一个节点移去；(3)将该节点固定在部分拓扑排序中。该过程中最大的部分是维持合格节点集合的内容，这项内容可以通过优先队列(priority queue)来有效地实现[134]。优先队列是用来有效存储和操作项目集合的抽象数据类型，集合中的每个项目都有一个与之相关联的值，称作关键字(key)或优先权(priority)。该数据结构支持将项目插入队列和从队列中将具有最高优先权的项目移除，即我们用来维持合格节点的操作。优先队列的一个特殊实现通过堆(heap)来完成。堆是一个具有特殊属性的二进制树，该树中每个节点的优先权都不小于其子节点的优先权[134]。堆可以存储在任何向量中，对于一个下标为 i 的节点，其父节点的下标为$\lfloor i/2\rfloor$，其左子节点的下标为 $2i$，右子节点的下标为 $2i+1$。假设合格节点集合包含 3,4,5,6,7 节点。每个节点所对应的优先权为 2,6,9,3,8。对应的用二进制树来表示的堆及其实现

向量见图 6.8。树中每个节点圈中的树是优先权值。节点旁边的数值是其在向量中对应的下标。

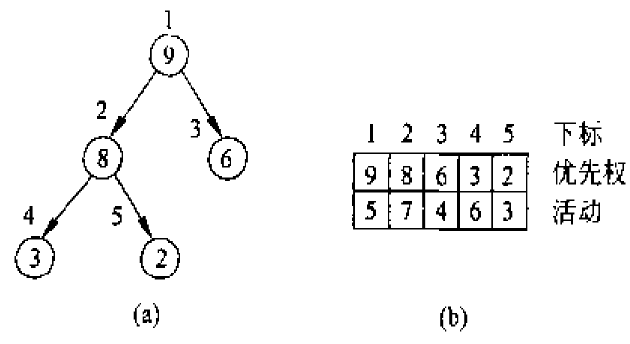


图 6.8 堆看作(a)二进制树;(b)向量

设 t 是过程的迭代下标,V 是所有节点的集合,Q_i 是所有活动 i 的直接紧前活动集合,PS[·]是存储拓扑排序的向量,CUT[i]是割集中传入节点 i 的边的数量,S_t 是第 t 步合格节点的集合。从染色体中产生拓扑排序的过程如下:

拓扑排序(topological sort)过程

第 1 步:初始化

$t \leftarrow 1$(迭代下标)

PS[t]$\leftarrow 1$(初始化拓扑排序)

$S_t \leftarrow Q_1$(初始化优先队列)

CUT[i]$\leftarrow 1, \forall i \in Q_1$(初始化割集中边的数量)

CUT[i]$\leftarrow 1, \forall i \in V - Q_1$

第 2 步:执行终止测试。如果 PS[t]$=n$,进入第 6 步,否则 $t \leftarrow t+1$,继续。

第 3 步:固定第 t 个节点,从优先队列 S_t 中移除具有最高优先权的节点 i^* 并将其放入向量 PS[t]。

第 4 步:执行割集更新

$$\text{CUT}[i] \leftarrow \text{CUT}[i] + 1, \qquad \forall i \in Q_{i^*}$$

第 5 步:更新合格节点集合。对于所有的 $i \in Q_{i^*}$,如果 CUT[i]$=d^{\text{IN}}(i)$,将 i 放入优先队列 S_t,返回第 2 步。

第 6 步:返回完整的拓扑排序 PS[·]。

6.4.2 遗传算子

遗传算法由遗传算子来实现并受到选择压力的导向。通常杂交算子用作主要算子,遗传系统的性能主要依赖于该算子;变异算子用于在不同的染色体中产生自发随机变化,因此成为次要算子。

在 Cheng 和 Gen 的实现中,采用了一种交互式方法来设计遗传算子:一个算子用于

执行广度搜索以探索局部最优以外的区域；另一个算子用于执行深度搜索以寻找改进的解。两种类型的搜索方法(深度和广度)构成了遗传搜索相互补充并完备的组成部分。采用这种方法时杂交和变异算子在遗传搜索中的作用相同。

基因位置的杂交(position-based crossover)　所提出的编码方法的本质可以看作一种排列编码。人们已经针对排列表示研究了许多重组算子。这里采用了 Syswerda 提出的基于位置的杂交算子[604]，如图 6.9 所示。这种方法随机地从一个父代中提取一些基因，子代中余下的基因通过对另一个父代进行从左到右的扫描并填充空位得到。

交换变异(swap mutation)　这里采用了交换变异算子，该方法简单地随机选择两个位置并交换其内容，如图 6.10 所示。

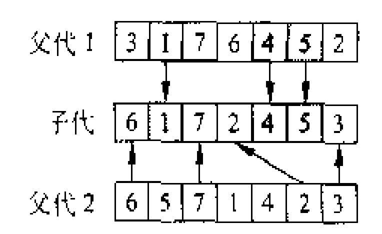


图 6.9　基于位置的杂交算子

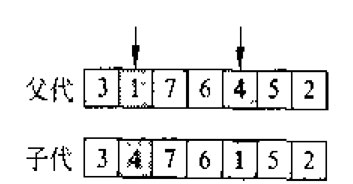


图 6.10　交换变异算子

基于局部搜索的变异(local search-based mutation)　局部搜索通过搜索当前解的邻域来寻找问题的改进解。局部搜索的实现需要初始当前解、对于初始当前解邻域的定义以及选择下一个当前解的方法。通过少量改变来寻找一个改进解的思想可以用于变异算子。观察交换变异，通过成对交换产生的染色体可以看作原始染色体的邻居。一个染色体的邻域(neighborhood)于是可以定义为通过成对交换产生的染色体集合。对于一对基因，一个称作中心(pivot)，该基因对于给定的邻域来说是固定的，另一个随机选择，如图 6.11 所示。在一个给定的邻域中，如果一个染色体在适应值上比其他任意染色体都要好，称其为局部最优(local optimum)。邻域的规模影响局部最优的质量。小规模邻域和大规模领域的折衷是清晰的：邻居数越大，找到优秀邻居的概率就越大，但寻找的时间也越长。这种变异算子的摘要描述如下：

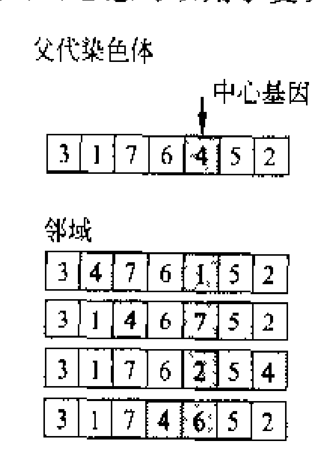


图 6.11　当前染色体及其邻域

基因局部搜索的变异过程

```
inputs: P(父代)
output: C(子代)
begin
    i←1;
```

```
    设 P 为当前染色体；
    选择一个中心基因；
    while i<特定的数 do
        随机选择一个基因；
        通过将其与中心基因交换产生一个邻居；
        评价这个邻居；
        如果这个邻居比当前染色体更优秀，将其设为当前染色体；
        i←i+1
    end
    保存 C 中的当前基因
end
```

6.4.3 评价与选择

在每一代中，采用某种适应值度量方式对染色体进行评价。在评价阶段采用了下面4个主要步骤：

第1步：将染色体转换为拓扑排序。

第2步：从拓扑排序产生调度。

第3步：计算每个调度的目标值。

第4步：将目标值转换为适应值。

第1步的过程在6.4.2节讨论。由于拓扑排序给出了活动的可行次序，我们根据活动在拓扑排序中出现的次序来构造调度，一旦资源允许就一次调度一个活动。设 i 为过程迭代的下标，V 为所有节点的集合，P_i 为活动 i 的所有直接紧前活动集合，PS[·]为存储拓扑排序的向量，σ_j 和 ϕ_j 分别为与活动 j 相关的开始和结束时间，$\boldsymbol{b}_k[l]$为记录时间 l 可行的资源 k 数量的向量，d_j 为活动 j 的持续时间，r_{jk} 为活动 j 对资源 k 的消耗。根据给定的拓扑排序确定每个活动开始和结束时间的过程如下：

活动的开始和结束时间过程

第1步：初始化

$i \leftarrow 1$(迭代下标)

$j \leftarrow \mathrm{PS}[i]$(初始化活动)

$\sigma_j \leftarrow 0, \phi_j \leftarrow 0$(初始化活动的开始和结束时间)

$$\boldsymbol{b}_k[l] \leftarrow b_k, l=1,2,\cdots,\sum_{j=1}^{n} d_j, k=1,2,\cdots,m \text{(初始化资源)}$$

第2步：执行终止测试。如果 $i=n$，进入第5步，否则 $i \leftarrow i+1$，继续。

第3步：确定开始和结束时间。

$j \leftarrow \mathrm{PS}[i]$

$\sigma_j^{\min} \leftarrow \max\{\phi_l \mid l \in P_j\}$

$\sigma_j \leftarrow \min\{t \mid t \geqslant \sigma_j^{\min},\ \boldsymbol{b}_k[l] \leqslant r_{jk}, l=t,t+1,\cdots,t+d_j,\quad k=1,2,\cdots,m\}$

$\phi_j \leftarrow \sigma_j + d_j$

第4步：更新可用资源

$\boldsymbol{b}_k[l] \leftarrow \boldsymbol{b}_k[l] - r_{jk},\quad l=t,t+1,\cdots,t+d_j,\quad k=1,2,\cdots,m$

返回第2步。

第5步：停止。返回 σ_j 和 ϕ_j。

由于目标是项目的持续时间，因此最后一个活动的结束时间就是目标值。我们处理的是最小化问题，因此必须将原始目标值转换为适应值以确保优秀个体具有大的适应值。

设 v_k 是当前种群的第 k 个染色体，$g(v_k)$是适应值函数，$f(v_k)$是目标值(即项目持续时间)，$f_{\max}$和 $f_{\min}$分别是当前种群的最大目标值和最小目标值。转换方法如下：

$$g(v_k) = \frac{f_{\max} - f(v_k) + \gamma}{f_{\max} - f_{\min} + \gamma} \tag{6.25}$$

其中 γ 是正实数，通常限制在开区间(0,1)中。使用 γ 的目的有2：(1)防止式(6.25)产生被零除；(2)使得可以将选择行为从适应值比例选择调整到纯随机选择。如果染色体间适应值的差距相对较大，则采用适应值比例选择；如果区别相对较小，则选择趋向于在相互竞争的染色体中进行纯随机选择。

在 Cheng 和 Gen 的试验中采用了**轮盘赌**(roulette wheel)方法(一种适应值比例选择)。**最优性选择**(elitist selection)与这种方法相结合在下一代中维持最优的染色体同时避免采样误差。在最优性选择中，如果当前代的最优个体没有在下一代中得到复制，随机从下一代中移除一个个体并将最优个体加入种群。

6.4.4　试验结果

初步的计算试验构造为两个部分：(1)调整遗传算法的参数以研究它们是如何影响性能的，(2)验证遗传搜索中基于局部搜索变异的性能。

参数设置　为了研究种群规模是如何影响算法性能的，将最大遗传代数固定为100，杂交率和变异率均固定在0.1。杂交率和变异率低时种群规模就成为遗传算法性能的主要影响因素了。种群规模从10到100变化。图6.12显示了每个参数设置下100次随机运行的最好、最差和平均目标值。从结果中可以看出，种群规模大于50以后，继续增加对于遗传算法的性能没有显著影响。

下面测试的目的是比较杂交和变异算子以研究哪个是遗传搜索中更为重要的算子。遗传算法在下面两种情况中进行了测试：

1. 固定变异率为0，将杂交率从0.1变化到0.9。

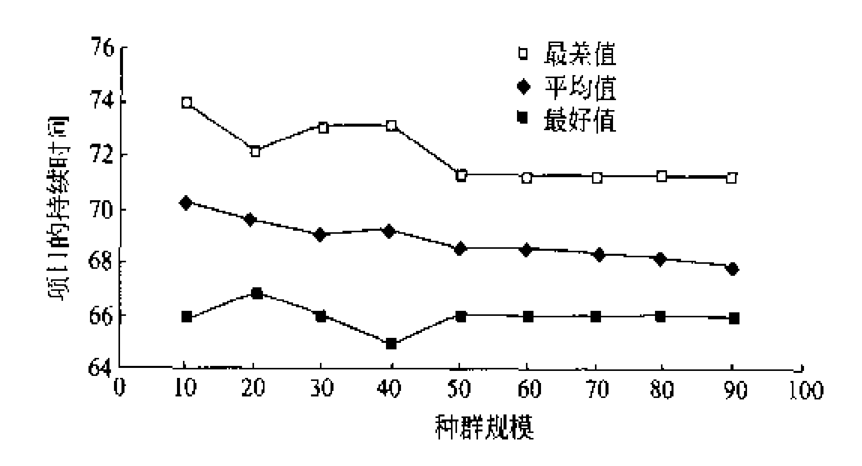


图 6.12 在不同 *pop_size* 值下目标值最好最差和平均值的比较

2. 固定杂交率为 0，将变异率从 0.1 变化到 0.9。

两种情况下均固定 *max_gen*=100，*pop_size*=20。图 6.13 给出了每个参数设置下 200 次随机运行的最好目标函数值，平均值见图 6.14，最差值见图 6.15。

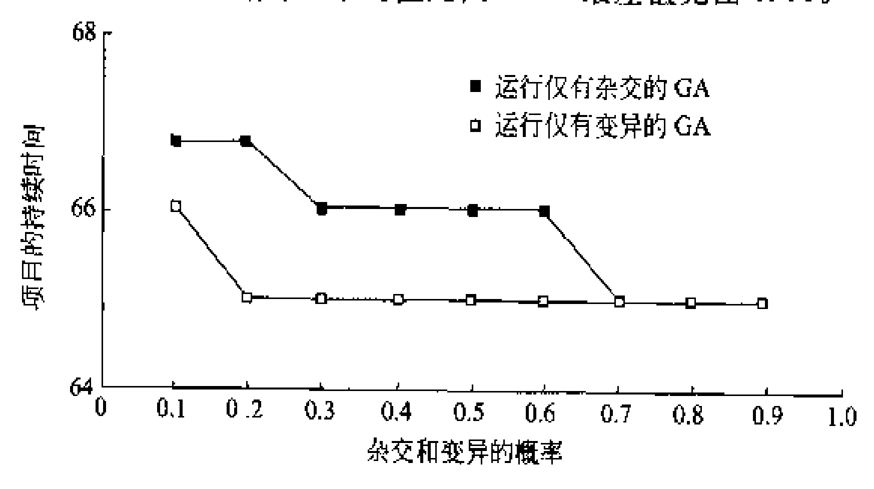


图 6.13 不同杂交率和变异率下 200 次随机运行最好值的比较

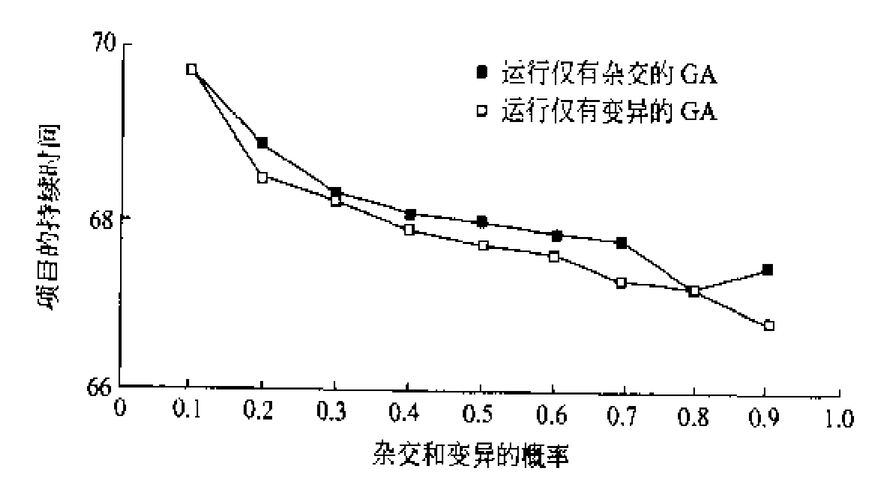


图 6.14 不同杂交率和变异率下 200 次随机运行平均值的比较

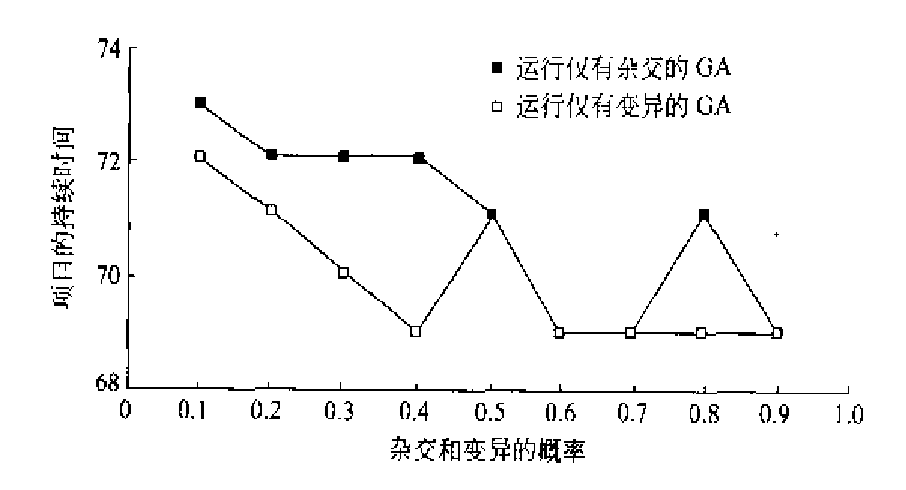


图6.15 不同杂交率和变异率下200次随机运行最差值的比较

从结果中可以看出对于平均值来说，杂交和变异之间没有明显的区别，但如果遗传算法的只有变异算子时获得最优解的机会要远大于只有杂交算子时的机会。这个结果表明变异在遗传中起决定性的作用，这与传统的观念相矛盾。在传统的遗传算法中，杂交作为主要算子，变异仅作为辅助方法。虽然交换变异的机制非常简单，但它允许该给定的染色体周围进行广度搜索。这就是变异能够以较大概率产生最优解的原因。

基于局部搜索的变异的行为 下面3种情况的试验验证了遗传搜索中基于局部搜索的变异的行为：(1)仅有基于位置的杂交；(2)仅有交换变异；(3)仅有基因局部搜索的变异。

为了对基因位置的杂交和交换变异进行公平的比较，种群规模固定为50，最大代数固定为200；对于基于局部搜索的变异，种群规模固定在20，最大代数为100。在每种情况下，杂交率和变异率均为0.3。基因局部搜索的变异的邻域规模固定为6，这样使得每种算子处理的染色体数量基本相同。仅采用交换变异时200次运行的解的分布见图6.16，仅采用基于位置的杂交时解的分布见图6.17，仅采用基于局部搜索的变异时解的分布见图6.18。容易看出基于局部搜索的变异对遗传算法的性能有明显的影响。

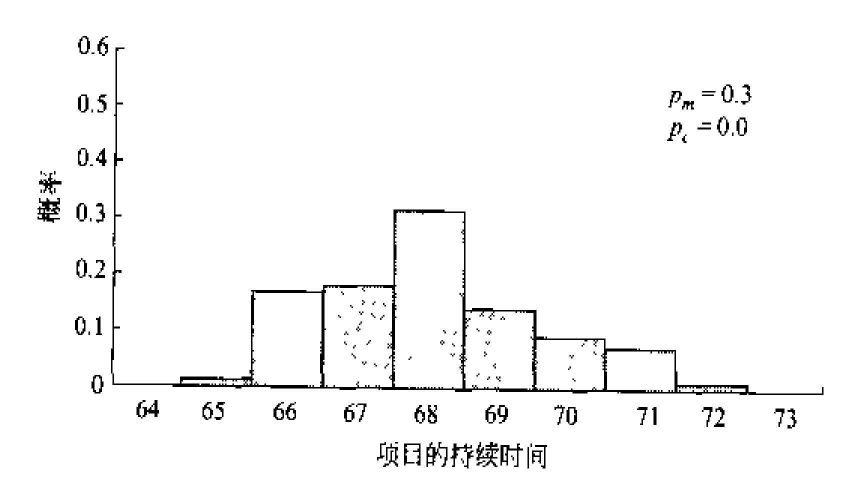


图6.16 仅有交换变异的200次运行的解分布

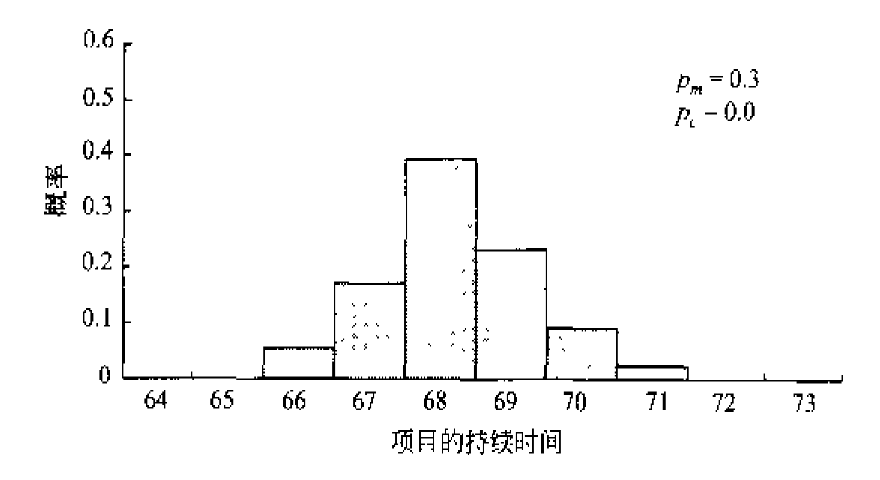


图 6.17 仅有基于位置杂交的 200 次运行的解分布

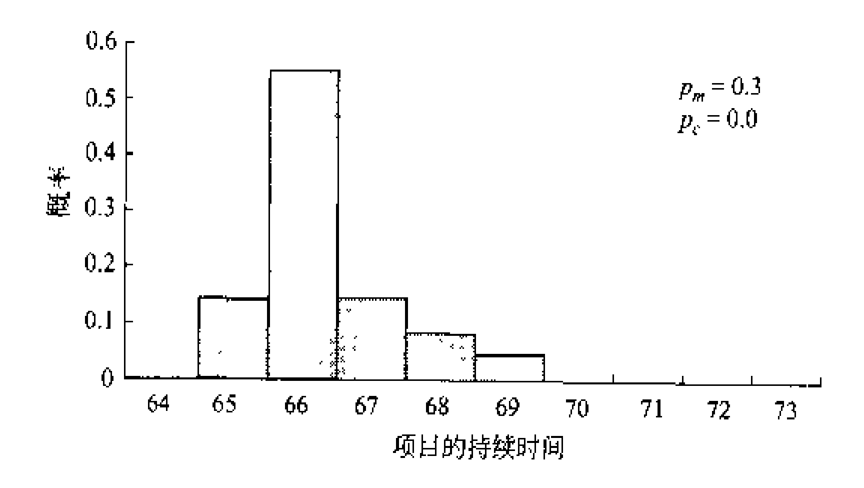


图 6.18 仅有基于局部搜索变异的 200 次运行的解分布

最近 Tsujimura 和 Gen 提出了用于求解资源约束环境下项目调度的进化算法[736]，Özdamar 提出了对于一般类别项目调度问题的遗传算法[727]。

6.5 并行机器调度

下面考虑并行机器调度(parallel machine scheduling)问题：

- 有 $m(m<n)$ 台相同的并行机器和 n 个具有已知权重 $w_1, w_2, \cdots, w_n$ 和加工时间 $p_1, p_2, \cdots, p_n$ 的独立作业。
- 作业随时可以进行加工，可以被 m 台机器中任意一台加工。
- 一旦加工开始，没有作业可以被占先(be preempted)。
- 对于所有作业都有公共的无限制的到期时间 d。

$$d \geqslant \sum_{j=1}^{n} p_j$$

容易验证,该问题的最优调度在作业之间没有空闲时间。设 Π 表示作业间没有空闲时间的可行调度集合。对于给定的调度 $\sigma\in\Pi$,设 c_j 为调度 σ 中作业 j 的完成时间($j=1,2,\cdots,n$),设 $f(\sigma)$ 表示对应的目标函数值。问题需要最小化最大的加权绝对延时:

$$\min_{\sigma\in\Pi} f(\sigma)=\max\{w_j\mid c_j-d\mid;\ j=1,2,\cdots,n\} \tag{6.26}$$

该目标被认为是一种不规则的性能度量。

通常假设所有的机器完全相同,因此作业的加工时间与机器无关。作业用加工时间和权重来表示其特征。每个作业一个时刻最多可由一台机器加工,一台机器在一个时刻只能加工一个作业。每个作业都有公共的到期时间。目标可以分为两类:**规则度量**(regular measure)和**不规则度量**(nonregular measure),后者一般比前者更难以求解。作为一个例子,考虑表6.4给出的9个作业3台机器问题。图6.19给出了一个可行调度。

表6.4 9个作业/3台机器的例子

作 业	1	2	3	4	5	6	7	8	9
加工时间	2	4	4	1	5	2	1	3	2
权重	5	8	6	4	4	1	9	10	2

已知处理并行机器调度问题存在两个基本的问题:(1)机器间的作业划分(job partition)(本质上是组合问题),(2)每台机器上的作业顺序(job sequence)(本质上是排列问题)。即问题中既包含了排列成分,又包含组合成分。大多数并行机器调度问题都是NP完全的[117]。

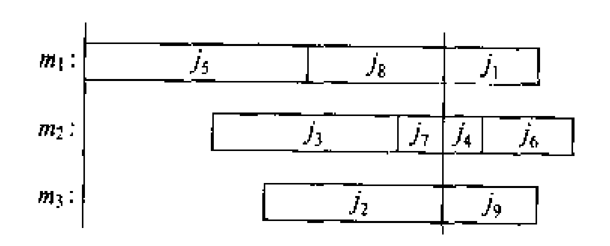


图6.19 9个作业3个机器例子的可行调度

Cheng和Gen研究了如何用memetic算法(一种遗传算法与局部搜索方法的混合算法)求解并行机器调度问题[100,104]。所有类型的并行机器调度问题都有两个基本问题:机器间的作业划分和每台机器上的作业顺序。在他们提出的方法中,遗传算法用来进化作业划分,启发式过程用来调整作业排列从而将每个染色体移动到对应的局部最优解上。作者建立了问题最优解的若干支配条件,并提出了一种基于支配条件的作业排序快速启发式算法。

6.5.1 支配条件

一种经常用来确定一台机器上作业次序的方法是在作业中建立某些支配性质(dominance property)。支配性质建立了最优调度中作业之间的前后关系。这通常称作后验(posteriori)前后关系,原因在于这些关系并非问题最初的表述。这种类型的前后关

系将被用来为机器上的作业顺序调度创建快速启发式算法。

设 J 是作业的集合。一个作业或是属于早作业集合(early job set),或是属于迟作业集合(tardy job set)。早作业集合的定义是 $E=\{j|c_j\leqslant d$ 且 $j\in J\}$,迟作业集合的定义是 $T=\{j|c_j>d$ 且 $j\in J\}$。对于给定的调度,如果一个作业具有最大加权绝对延时,则该作业称作支配作业(dominant job)。如果作业 i 是支配作业,则 $w_i|c_i-d|=\max\limits_{j\in J}\{w_j|c_j-d|\}$。如果一台机器加工支配作业,则该机器称作支配机器(dominant machine)。

性质 6.4 对于迟作业集合 T 中的一对作业 i 和 j,如果 $w_i\geqslant w_j$,作业 i 至少在一个最优调度中领先作业 j。

证明 考虑图 6.20 所示两种可能的作业 i 和 j 的次序,其中 x 表示迄今为止集合 T 中已调度的作业的总长度。我们有:

$$\frac{w_i(p_i+p_j+x)}{w_j(p_i+p_j+x)}=\frac{w_i}{w_j}\geqslant 1 \tag{6.27}$$

因此作业 i 一定领先作业 j。

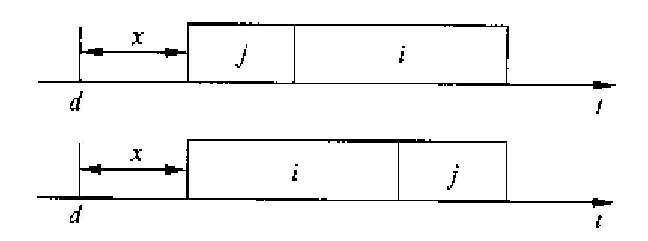


图 6.20 迟作业集合中作业 i 和 j 的两种可能次序

这个性质揭示了这样一个事实,迟作业集合 T 中至少一个最优调度有一个权重大的作业领先一个权重小的作业。换句话说,集合 T 中的作业按照权重的非增次序排列。

早作业集合 E 中作业的次序不像迟作业集合 T 中那样简单。一个作业由两方面的因素来表征:权重和加工时间。给定两个作业 i 和 j,其次序关系有 4 种基本模式:

1. $w_i\geqslant w_j$ 且 $p_i\geqslant p_j$
2. $w_i\geqslant w_j$ 且 $p_i<p_j$
3. $w_i<w_j$ 且 $p_i\geqslant p_j$
4. $w_i<w_j$ 且 $p_i<p_j$

对于给定作业对的前后关系来说,模式 4 描述了和模式 1 相同的情况,模式 3 和模式 2 的情况相同,因此我们仅需验证模式 1 和模式 2。

性质 6.5 对于早作业集合 E 中的一对作业 i 和 j,如果 $w_i\geqslant w_j$ 而且 $p_i<p_j$,则作业 j 至少在一个最优调度中领先作业 i。

证明 考虑图 6.21 所示的两种可能的作业 i 和 j 的次序,其中 x 表示迄今为止集合 E 中已调度的作业的总长度。我们有:

$$\frac{w_i(p_j+x)}{w_j(p_i+x)} \geqslant \frac{w_i(p_i+x)}{w_j(p_i+x)} = \frac{w_i}{w_j} \geqslant 1 \tag{6.28}$$

因此作业 j 领先作业 i。

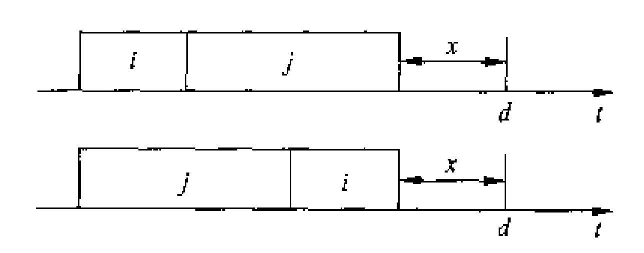


图 6.21　早作业集合中作业 i 和 j 的两种可能次序

该性质表明，早作业集合 E 中至少一个最优调度有加工时间长而且权重小的作业领先加工时间短而且权重大的作业。

性质 6.6　对于早作业集合 E 中的一对作业 i 和 j，如果 $w_i \geqslant w_j$，$p_i \geqslant p_j$，而且 $w_i/p_i \geqslant w_j/p_j$，则作业 j 领先作业 i。

证明　考虑图 6.22 所示的两种可能的作业 i 和 j 的次序。我们有：

$$\frac{w_i(p_j+x)}{w_j(p_i+x)} \geqslant \frac{w_i p_j + w_j x}{w_j(p_i+x)} \geqslant \frac{w_j p_i + w_j x}{w_j(p_i+x)} = 1 \tag{6.29}$$

因此作业 j 一定在一个最优调度中领先作业 i。

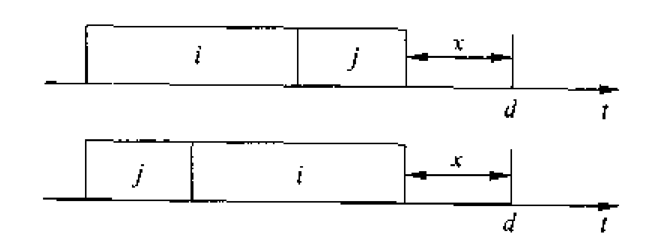


图 6.22　早作业集合中作业 i 和 j 的两种可能次序

该性质给出了在早作业集合 E 中至少一个最优调度有加工时间短而且权重小的作业领先加工时间长而且权重大的作业的必要条件之一。

性质 6.7　如果集合 E 中的一对作业 i 和 j 有 $w_i \geqslant w_j$，$p_i \geqslant p_j$，以及 $w_i/p_i < w_j/p_j$，而且如果集合 E 中已调度作业的总长度 x 满足

$$x \geqslant \frac{w_j p_i - w_i p_j}{w_i - w_j} \tag{6.30}$$

则作业 j 至少在一个最优调度中领先作业 i。

证明　考虑图 6.22 所示的两种可能的作业 i 和 j 的次序。如果集合 E 中作业 j 领先作业 i，我们有

$$\frac{w_i(p_j+x)}{w_j(p_i+x)} \geqslant 1 \tag{6.31}$$

这意味着

$$x \geqslant \frac{w_j p_i - w_i p_j}{w_i - w_j} \tag{6.32}$$

该性质给出了最优调度中加工时间短而且权重小的作业领先加工时间长而且权重大的作业的另一个必要条件。从前述性质我们得知至少存在一个最优调度，大多数情况下该调度在支配机器上集合 T 中的作业按照权重的非降次序排列，在支配机器上集合 E 中的作业按照权重的非增次序排列。如果权重小而且加工时间长的作业不满足性质 6.7 中的条件，则会发生例外。

性质 6.8 对于一个给定的最优调度，在支配机器上存在两个支配作业，一个在集合 T 中，另一个在集合 E 中。

证明 假设对于给定的最优调度，只有一个支配作业。不失一般性，假设作业 $i \in E$ 是支配作业。于是一定存在一个支配集合 T 的作业 $j \in T$，而且 $w_j|c_j-d|<w_i|c_i-d|$，如图 6.23 所示。如果将所有作业都延迟数量 δ，δ 由下式确定：

$$\delta = \frac{w_i \mid c_i - d \mid - w_j \mid c_j - d \mid}{w_i + w_j} \tag{6.33}$$

则两个作业 i 和 j 具有相同的绝对延时值，给定调度的目标值减少了 $w_i\delta$。这与最优调度的前提条件相矛盾。因此证明了性质。

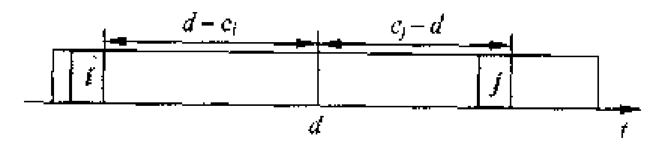


图 6.23 最优调度中的两个支配作业 i 和 j

性质 6.9 对于支配机器来说至少存在一个最优调度满足下面的关系：

$$\sum_{i \in E} p_i \geqslant \sum_{j \in T} p_j \tag{6.34}$$

证明 在一个支配机器上支配作业有下面 4 种可能的模式。

1. 一个作业在 E 最前面而另一个在 T 的最后面。
2. 一个作业在 E 最前面而另一个在 T 的中间。
3. 一个作业在 E 中间而另一个在 T 的最后面。
4. 一个作业在 E 中间而另一个在 T 的中间。

对于情况 1 和 3(如图 6.24 所示)，假设 T 的总长度比 E 的总长度长。于是可以通过将 T 中最右边部分移动到 E 中最左边部分来活动更好的调度，这使得给定调度不再是最优的。这与最优调度的前提条件相矛盾。对于图 6.24 所示的情况 2 和 4，不失一般性，假设如果最右作业 k 和 m 从迟作业集合 T 中移出，两个集合的长度几乎相同。这种情况下可以通过将作业 m 放到 E 最左边，作业 k 放到作业 m 的后面来创建一个新的调度。这样 E 的总长度要长于 T，同时最大绝对延时也没有增加，于是证明了性质。

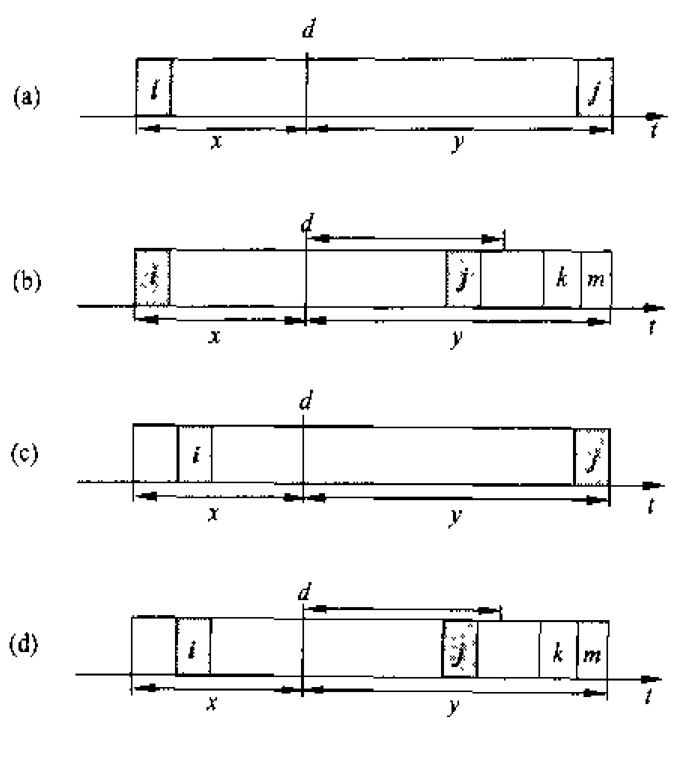


图 6.24　支配作业的4种可能模式

6.5.2 memetic 算法

memetic 算法(memetic algorithms)可以看作一种在局部最优子空间上进行的特殊类型遗传搜索,在遗传算法中加入了局部优化方法并将其应用于每个个体(在个体被插入到种群之前)[468,520]。重组和变异通常会产生不在当前局部最优子空间中的解,但局部优化方法可以修补这些解,产生的最终解维持在当前子空间中,这就是 memetic 算法。在这种混合的方法里,遗传算法用于进行种群中的全局广度搜索,而启发式方法则用于进行染色体间的局部深度搜索。由于遗传算法和传统启发式方法具有互补性质,这种混合方法的性能通常比单独运行任意一种要好。

表示　仅需将作业划分部分编码入染色体。设 $J=\{1,2,\cdots,n\}$ 表示作业集合,$M=\{1,2,\cdots,m\}$ 表示机器集合。一条染色体包含 n 个基因,每个基因从集合 M 中选取一个整数。于是基因的位置表示作业的 ID,基因的值表示机器的 ID。图 6.25 用 3 台机器 9 个作业的简单例子说明了这种编码方法。在这个例子中,作业 2,3,9 在机器 1 上加工,作业 6 和 7 在机器 2 上加工,作业 1,4,5,8 在机器 3 上加工。

遗传算子　这里采用了Syswerda提出的均匀杂交算子[603]。该方法首先产生一个

1	2	3	4	5	6	7	8	9	位置：作业 ID
3	1	1	3	3	2	2	3	1	值：　机器 ID

图 6.25　提出的编码方法

随机的掩码，然后根据掩码交换父代中对应的基因。杂交掩码就是与染色体长度相同的二进制串。掩码中每位的奇偶性决定了后代中该位从哪个父代所对应的位继承。容易看出杂交算子可以在机器间调整作业划分。

观察图 6.26 中第二个子代可以知道机器 3 从染色体中消失了。这意味着没有作业分配给机器 3。这种消失现象显然会产生一个糟糕的调度。为了让遗传搜索能够在染色体中恢复一个机器，使用了随机替换(random replacement)变异算子。该变异算子首先随机选择一个基因，然后将该基因的值替换为集合 M 中随机选出的一个整数，如图 6.27 所示。该变异算子本质上既可导致一台基因的恢复，也可导致基因的消失。该算子可以在机器间调整作业的划分。

启发式过程 启发式过程用于对每台机器上的作业进行排序，排序的原则是性质 6.4～6.7给出的支配条件。

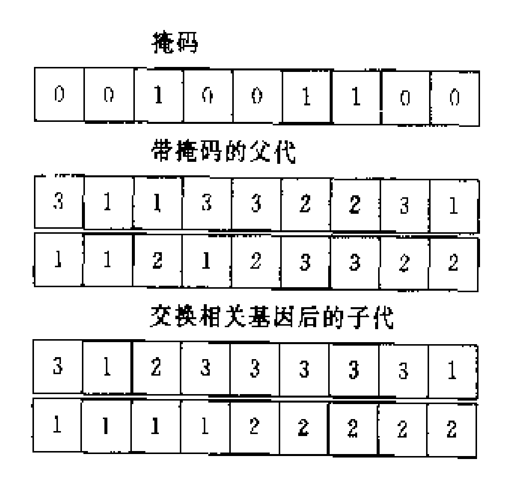


图 6.26 均匀杂交操作

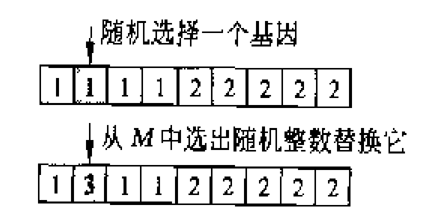


图 6.27 随机替换变异操作

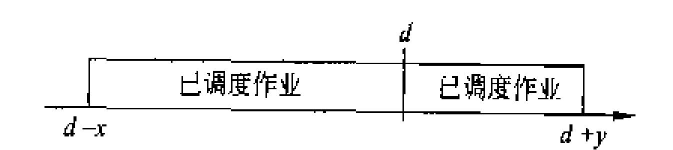


图 6.28 已调度作业的开始和完成时间

设 $d-x$ 表示一台机器上已被调度的最早作业的开始时间，$d+y$ 表示一台机器上已被调度的最迟作业的完成时间，如图 6.28 所示。设 J 为作业集合，E 为早作业集合，$E[i]$ 为 E 中第 i 个元素，T 为迟作业集合。启发式过程如下：

作业排序过程

输入：一台机器上的作业集合 J
输出：集合 E 和集合 T 中的作业顺序
begin

    根据权重的非增顺序对 J 中所有作业进行排序；
    将排序后的作业保存入 S[·]；
    $x \leftarrow 0$，$y \leftarrow 0$；
    $i \leftarrow 0$；
    while $i \leqslant |J|$ **do**
        $p \leftarrow p_{S[i]}$；
        if $y+p>x$

```
            then 将作业 i 放到集合 E_t 的最前端;
            x ← x + p;
            if p_E[2] ≥ p_E[1] 而且 w_E[1] p_E[2] ≥ w_E[2] p_E[1] 而且
               y < (w_E[1] p_E[2] − w_E[2] p_E[1]) / (w_E[2] − w_E[1]) − p_E[2]
            then 交换 E[1]和 E[2];
            end
        else
            将作业放入集合 T 的尾端;
            y ← y + p;
        end
        i ← i + 1;
    end
end
```

评价与选择 两个主要步骤描述如下:

1. 计算每个染色体的目标值。
2. 将目标值转换为适应值。

从式(6.26)可以看出目标是作业完成时间的函数。完成时间(开始时间)本质上可以根据上面介绍的启发式过程所得到的次序计算出来。然而如果用这种方式计算完成时间,在支配机器上仅能产生一个支配作业。根据性质6.8,我们可以适当地滑动每个作业以产生更好的调度。因此对于给定的染色体,我们首先根据式(6.33)在每台机器上滑动每个作业,然后计算染色体的最大绝对延时目标值。

由于是最小化问题,必须将原始目标值转换为适应值以确保优秀的个体具有大的适应值。这种转换通过下面介绍的分数线性函数来完成:

$$\mathrm{eval}(v_t) = \frac{1}{f(v_t)}, \quad t = 1,2,\cdots,pop_size \tag{6.35}$$

其中 $\mathrm{eval}(v_t)$ 是第 t 个染色体的适应值,$f(v_t)$ 是其目标值。

采用了轮盘赌选择(roulette wheel selection)作为基本选择方式,该方法从当前扩大的种群中复制下一代。将轮盘赌方法与最优性选择相结合以在下一代中保持最优染色体并克服采样的随机误差。

6.5.3 试验结果

在随机产生的测试问题上进行了初步的计算试验。首先用不同的参数设置来研究该算法以研究这些参数是如何影响算法性能的。另外还与 Li 和 Cheng 提出的启发式算法进行了比较以验证所提出算法的有效性。

参数调整 研究了种群规模是如何影响 memetic 算法性能的。采用了随机产生的

30 个作业和 5 台机器的问题进行测试，固定 max_gen 为 500，p_m 和 p_c 为 0.1。杂交率和变异率较低时种群规模就成为影响 memetic 算法性能的主要因素。图 6.29 显示了 pop_size 从 10 变化到 90 时 100 次随机运行得到的最好、最差和平均目标值。从结果可以看出种群规模大于 50 以后，继续增加对性能没有明显影响。

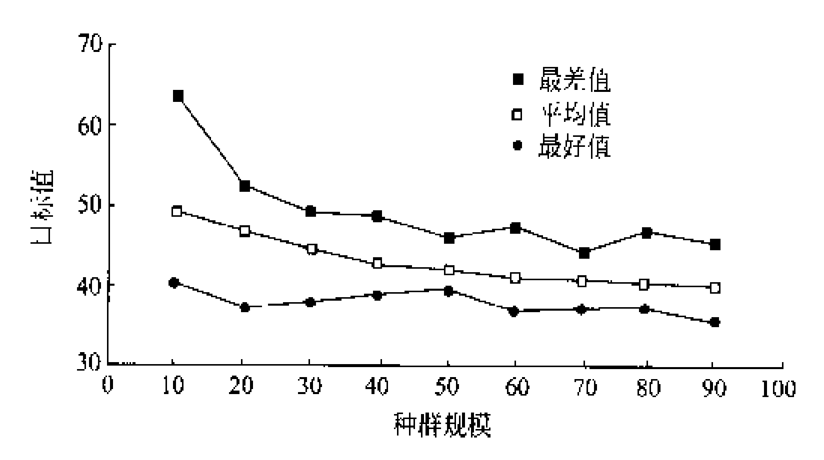


图 6.29 memetic 算法通过改变 pop_size 得到的目标值的最好、最差和平均值比较

与启发式算法比较 采用了随机产生的具有不同作业和机器规模的问题进行比较。memetic 算法基本的参数设置为：$pop_size=50$，$p_c=0.4$，$p_m=0.4$，$max_gen=500$。对每个问题进行 20 次运行的结果如表 6.5 所示，其中启发式算法即 Li 和 Cheng 提出的方法。结果表明，memetic 算法好于启发式算法。

表 6.5 随机测试问题的结果

问题	规　　模	启发式	memetic 算法		
			最好	最差	平均
1	$j30\times m5$	38.50	32.00	35.20	33.52
2	$j50\times m5$	100.15	94.50	102.86	98.59
3	$j50\times m10$	49.09	38.77	43.31	40.91
4	$j60\times m5$	117.78	114.55	120.00	117.37
5	$j60\times m10$	132.00	125.36	140.00	133.59
6	$j70\times m8$	88.00	84.00	93.17	87.06
7	$j70\times m10$	150.86	142.22	156.80	147.88
8	$j80\times m5$	292.50	284.39	304.00	292.43
9	$j80\times m10$	97.07	92.00	99.47	95.75
10	$j80\times m12$	98.18	92.31	97.78	94.72
11	$j80\times m14$	81.67	73.85	82.13	76.84
12	$j90\times m7$	150.00	145.38	151.20	148.47
13	$j90\times m11$	77.50	71.11	78.24	74.05
14	$j100\times m20$	113.75	106.36	116.31	110.61

6.6 多处理器调度问题

多处理器系统广泛地在计算机应用中得到采用。由于多处理器系统能够提供很好的响应和投人产出比，其应用的领域不仅是信息处理，还包括机器人控制、实时高速动态系统仿真等等。

多处理器调度问题(multiprocessor scheduling problems，MSPs)可以描述为将具有前后约束任务图中的任务分配到处理器集合上，确定每个处理器任务执行的顺序以最小化某种费用函数，如整体任务执行(完成)时间。然而这种问题一般来说是很难处理的。众所周知，从该问题的原始问题经过松弛和简化得到的子问题在添加了若干限制条件后仍然属于NP-难问题。

Huo，Ren和Ansari对于这种问题开发了可行的遗传编码[308]。他们的基因型被划分为可变长度的分割或子染色体。Yue和Lilja提出了一种并行分布式遗传算法[688]。运行于本地工作站的简单遗传算法用于产生可能的调度算法。Tsujimura和Gen为求解该问题提出了基于高度函数的遗传算法[627]。

6.6.1 问题描述与假设

多处理器调度就是将 n 个任务分配到 m 个处理器上(需要满足前后约束(precedence constraints))，确定每个任务的开始和结束时间，目标是最小化完成时间。问题的数学表达如下：

$$\min \quad \{\max_j\{x_j y_{ij}\}\}$$

$$\text{s.t.} \quad x_k - p_k \geqslant x_j, T_j \prec T_k \tag{6.36}$$

$$t_{\max} - \sum_{j=1}^{n} p_j y_{ij} \geqslant 0, \quad i = 1,2,\cdots,m \tag{6.37}$$

$$\sum_{i=1}^{m} y_{ij} = 1, \quad j = 1,2,\cdots,n \tag{6.38}$$

$$y_{ij} = 0 \text{ 或 } 1, \quad i = 1,2,\cdots,m, \quad j = 1,2,\cdots,n \tag{6.39}$$

其中如果任务 T_j 分配给处理器 P_i，则 $y_{ij}=1$，否则为0。另外 $t_{\max}=\max_i\{t_i\}$，x_j 是任务 T_j 的完成时间，p_j 是任务 T_j 的处理时间，t_i 是处理分配给处理器 P_i 所有任务所需的时间，$\prec$表示了前后关系，$T_j \prec T_k$ 意味着 T_k 领先 T_j。假设通信系统无冲突并在所有数据获得以后才允许重叠通信。相同任务的复制是不允许的。

6.6.2 求解MSP的遗传算法

对于染色体表示方法和遗传操作，我们采用了高度函数(height function)的概念[308]，在遗传算法的实现上考虑了任务间的前后关系。遗传算子的构造基于任务的高

度函数。基于高度函数提出了3种遗传算子。

高度函数(height function) 为了促进调度的产生和遗传算子的构造，我们定义任务图中每个任务的高度为：

$$height(T_i)=\begin{cases}0, & \text{如果 } \mathrm{pred}(T_i)=\varnothing \\ 1+\max\limits_{T_j\in \mathrm{pred}(T_i)}\{height(T_j)\}, & \text{否则}\end{cases}$$

其中 $\mathrm{pred}(T_i)$是 T_i 前面的任务集合，$\mathrm{succ}(T_j)$是 T_j 后面的任务集合。这种定义一个主要的缺点在于最优调度可能不满足该定义。为了减少这种情况发生的可能性，我们可以修改高度的定义为：

$$height'(T_j)=\mathrm{rand}\in[\max\{height(T_i)\}+1,\min\{height'(T_k)\}-1]$$
$$\text{在所有 } T_i\in \mathrm{pred}(T_j),\ T_k\in \mathrm{succ}(T_j)\text{ 上} \tag{6.40}$$

其中 rand 是随机整数。这个高度函数在某种程度上代表了任务间的前后关系。事实上，如果有从 T_i 到 T_j 的路径(即存在一连串有向边从 T_i 指向 T_j)，则 T_i 一定在 T_j 之前执行，$height(T_i)<height(T_j)$。然而如果两个任务之间没有路径，它们的执行次序可以是任意的。但必须满足处理器内的前后关系(precedence relations)。

例 6.1 考虑一个有 8 个作业和 2 个处理器的实例。每个任务的高度如图 6.30 所示。

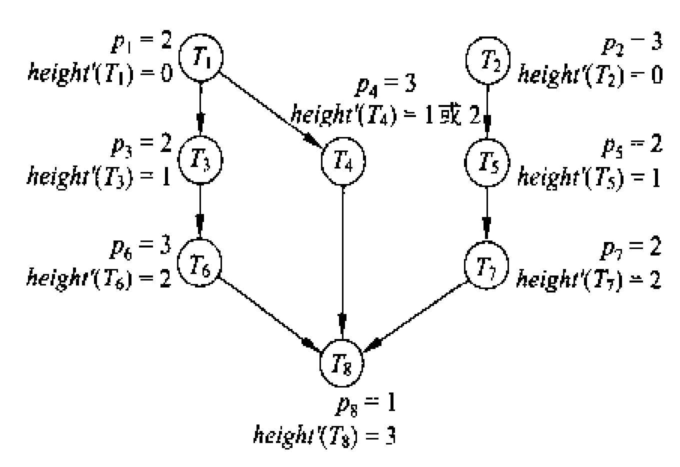


图 6.30 例 6.1 的任务图

表示 这里采用的染色体表示基于每个处理器中的任务调度。这种表示不需要考虑调度到不同处理器的任务间的前后关系。遗传算法对调度的表示必须包含计算任务之间的前后关系。这个要求通过将调度表示成若干计算任务的列表来实现。每个列表代表一个处理器上执行的计算任务，列表中任务的次序表示其执行次序。这个次序可以让我们维持在一个处理器上执行的任务的前后关系(处理器内优先关系(intraprocessor precedence relation))并忽略在不同处理器上执行的任务的前后关系(处理器间优先关系(interprocessor precedence relation))。

为了简化遗传算子的构成，我们在调度上加入了下面的条件：每个处理器上调度的任务列表按照其高度的升序排列。

例 6.2 考虑图 6.30 所示的 MSP。包含两个列表 P_1 和 P_2(对应两个处理器)的染色体如下：

$P_1(T_1T_5T_4T_7)$：$height'(T_1) \leqslant height'(T_5) \leqslant height'(T_4) \leqslant height'(T_7)$

$P_2(T_2T_3T_6T_8)$：$height'(T_2) \leqslant height'(T_3) \leqslant height'(T_6) \leqslant height'(T_8)$

Gantt 图参见图 6.31。这个调度的染色体 v 表示如下：

$$v = \begin{cases} P_1(T_1T_5T_4T_7) \\ P_2(T_2T_3T_6T_8) \end{cases}$$

图 6.31 两个处理器的调度

评价函数 评价函数本质上就是问题的目标函数。该函数提供了一种评价搜索节点和控制选择过程的方法。对于 MSP 我们考虑吞吐量、完成时间和处理器利用率等因素作为评价函数。

这里提出的应用于遗传算法的评价函数基于调度的 C_{max}。由于我们采用轮盘赌选择，于是使用的评价函数为：

$$\text{eval}(v_k) = \frac{1}{C_{max}^k}, \quad k = 1,2,\cdots,pop_size \tag{6.41}$$

其中 C_{max}^k 表示第 k 个染色体的目标函数值，pop_size 是种群规模。

遗传算子 遗传算子的功能是基于当前种群的搜索节点创造新的搜索节点。新的搜索节点一般通过组合与重新排列旧搜索节点的一部分来构成。对于 MSP，这里采用的遗传必须要强调处理器内前后关系和调度中任务的完整性及惟一性，这样才能确保新产生的染色体总是代表合法的搜索节点。Tsujimura 和 Gen 采用轮盘赌选择(roulette wheel selection)，并将图 6.30 显示的任务图作为一个例子[627]。

算子 1 算子 1 根据下面步骤执行：

第 1 步：在区间$[1, \max\{height'\}]$中产生随机数 c。

第 2 步：在每个处理器上放置分割点，使得分割点前面任务的高度小于 c，而分割点后面的任务高度则大于或等于 c。于是我们在每个处理器上获得了两个部分调度。

第 3 步：交换第二个部分调度。

例 6.3 $c=2$; $height'(T_6)$, $height'(T_7)$, $height'(T_8) \geqslant 2$

$$v=\begin{cases}P_1(T_1T_5T_4\mid \mathbf{T}_7)\\P_2(T_2T_3\mid \mathbf{T}_6\mathbf{T}_8)\end{cases}$$

$$v'=\begin{cases}P_1(T_1T_5T_4\mid \mathbf{T}_6\mathbf{T}_8)\\P_2(T_2T_3\mid \mathbf{T}_7)\end{cases}$$

算子 2 算子 2 根据下面步骤执行:

第 1 步:在区间$[1,\max\{height'\}]$中产生随机数 c。

第 2 步:在每个处理器上选出 $height'$等于 c 的所有任务。

第 3 步:随机替换所有任务的位置。

例 6.4 $c=1;height'(T_3),height'(T_4),height'(T_5)=1$

$$v=\begin{cases}P_1(T_1\mathbf{T}_5\mathbf{T}_4T_6T_8)\\P_2(T_2\mathbf{T}_3T_7)\end{cases}$$

$$v'=\begin{cases}P_1(T_1\mathbf{T}_3T_6T_8)\\P_2(T_2\mathbf{T}_4\mathbf{T}_5T_7)\end{cases}$$

算子 3 我们在产生初始种群时,因为范围很宽,因此对每个任务固定 $height'$,这个值在所有代中都不改变。为了消除这种不便,Tsujimura 和 Gen 提出了第三种算子。该算子的目的是建立更大的搜索空间以寻找更好的染色体[627]。算子 3 根据下面的步骤执行:

第 1 步:选择那些 $height'$可以改变的任务。

第 2 步:用小概率在这些任务中随机选择。

第 3 步:改变选出任务的 $height'$值为该任务可以取的值,并根据修改后的 $height'$将其移动到合适的位置。

例 6.5 选择任务 T_4 作为待改变的任务(即任务 T_4 的 $height'$可以从 1 改变为 2)。改变以后交换任务 T_4 和 T_5 的位置。

$$height'(T_4)=1\rightarrow 2$$

$$height'(T_5)=1$$

$$v=\begin{cases}P_1(T_1T_3T_6T_8)\\P_2(T_2\mathbf{T}_4\mathbf{T}_5T_7)\end{cases}$$

$$v'=\begin{cases}P_1(T_1T_3T_6T_8)\\P_2(T_2\mathbf{T}_5\mathbf{T}_4T_7)\end{cases}$$

6.6.3 数值例子

在一个数值例子中,Tsujimura和Gen(T-G)采用了Stanford操作图[347]中的88个任务作为大规模MSP,如图6.32所示。在图6.32中不需要考虑节点1和90,原因在于

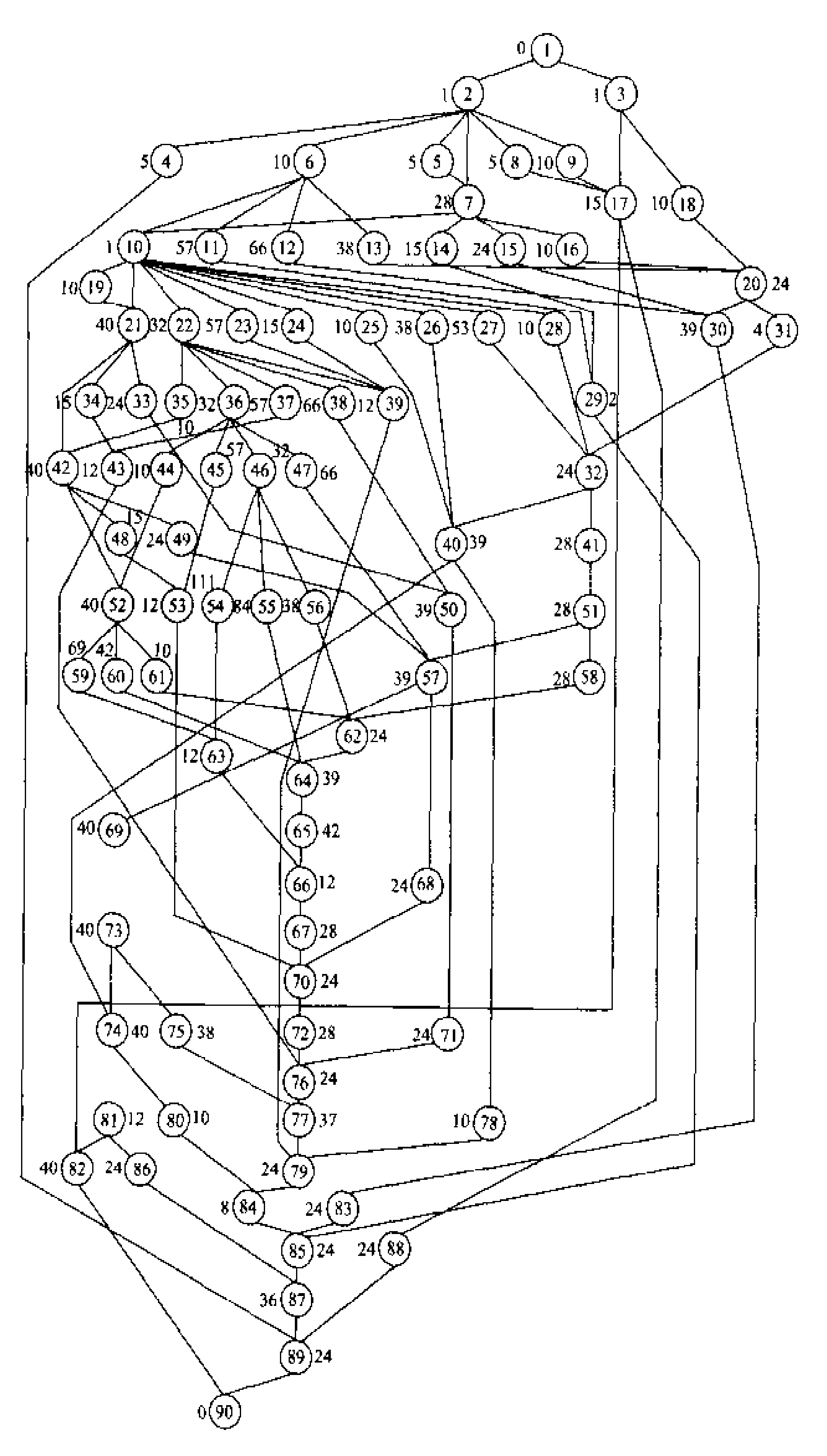


图 6.32 测试问题的可行调度

它们是附加的虚拟节点。设置为 $pop_size=30$，$gen_max=2000$。遗传操作概率 p_1，p_2，p_3，p_4 根据试验设置为合适的值，如表 6.6 所示。

表 6.6 遗传算法参数设置

处理器数量	p_1	p_2	p_3	p_4
2	0.6	0.3	0.5	0.1
4	0.7	0.3	0.3	0.1
6	0.7	0.3	0.4	0.05
10	0.6	0.3	0.5	0.1

不同处理器数量的比较结果见表 6.7。每个结果都是 20 次运行的最佳结果。表 6.7 表明结果比最优解大 0.0～13.2%，Tsujimura 和 Gen[627]提出的遗传算法比 Hou 提出的遗传算法[308]更有效。比较结果验证了提出的遗传算法可以在大规模 MSP 上得到良好的解。

表 6.7 比较结果

处理器数量	T-G 的 GA	Hou 等人的 GA[295]	最优解[295]
2	1247	1246	1242
4	759	774	659
6	619	627	573
10	570	590	570

最近 Hadj-Alouane，Bean 和 Murty[720]针对处理器任务分配问题开发了一种混合遗传/最优化算法，还对于多选择整数规划问题用遗传算法进行了讨论[719]。

第7章 高级运输问题

7.1 引言

运输问题(transportation problem)是一个基本的网络问题。许多学者提炼和扩展了基本的运输模型，不仅包括最优运输模式的确定，还包括生产计划(production scheduling)、转运问题和指派问题的分析。近来，有许多有关解决这类运输问题的进化方法的研究。Michalewicz等人[455,457]首先讨论了使用遗传算法来解决线性和非线性运输问题，使用矩阵来构建染色体的表示。他们在研究中设计了基于矩阵的杂交和变异。Gen和Li[234,412]讨论了用生成树表示的求解运输问题的遗传算法。他们使用树型编码的Prüfer数(Prüfer number)，作为设计染色体的可行性标准。使用生成树编码的优点在于：大大减少了设计染色体所需的存储单元。在m个工厂n个仓库的问题中，对于每一染色体，基于矩阵的表示法在进化过程中需要$m\times n$个存储单元，而使用Prüfer数表示法，仅需要$m+n-2$个存储单元。基于矩阵的染色体表示在执行中需要更多的存储单元，在解决问题时就需要更多的时间。

7.1.1 运输模型

生产计划和库存控制 多阶段的生产计划和库存控制(inventory control)可以构造成运输问题。典型的例子是季节性销售模式的生产商，但是生产保持完全的稳定，或生产的变化能满足销售模式(没有库存)，或生产能够在这两个极端情况之间采用一些折中的方法。

最直观的问题是加班费用和存储成本之间的平衡。在每个时间周期，比如说一个月，我们规定单位时间的正常和加班能力，例如：生产时数。我们也估计正常和加班生产的单位成本，以及每一周期单位库存成本。运输模型视每个周期内正常和加班的生产能力为源点(origins)，每个周期的需求为终点(destinations)。决策变量x_{ij}被定义为在给定周期内通过给定方式(正常或加班)生产出来的本期或未来交货的产品数量。与每个单元都相关的成本c_{ij}被定义为单位生产成本和单位存储成本的总和。

例如，考虑一个4周期的模型，周期1、2、3、4的预计需求分别为40、60、75和60个单位。正常时间单位成本是4，加班时间单位成本是6，每个周期单位存储成本是1.5。为了简化，我们假设每个周期的成本相同。而且在每个周期，正常生产能力是50个单位，加班生产能力是20个单位。表7.1给出了这个例子的运输表。

表 7.1　运输表

源点（生产源）	终点（周期需求）					供应（生产能力）
	1	2	3	4	虚构	
1. 正常周期 1	4.0	5.5	7.0	8.5	—	50
2. 加班周期 1	6.0	7.5	9.0	10.5	—	20
3. 正常周期 2	—	4.0	5.5	7.0	—	50
4. 加班周期 2	—	6.0	7.5	9.0	—	20
5. 正常周期 3	—	—	4.0	5.5		50
6. 加班周期 3	—	—	6.0	7.5	—	20
7. 正常周期 4	—	—	—	4.0	—	50
8. 加班周期 4	—	—	—	6.0	—	20
需求	40	60	75	60	45	280

因为总供应量大于总需求量，故建立一虚拟终点接受过剩产品。在这种情况下，每个虚拟点代表未使用的生产能力。虚拟点的容量一律赋为 0 成本。可以推断维持正常生产成本的能力是完全可变的。换言之，如果生产成本 = 直接劳动力 + 原材料 + 日常开支，则虚拟成本为 4。这个问题可以如 7.1.2 节那样构造为运输模型。

生产计划问题　生产计划问题将决定机器的生产以使总加工成本最低化。表述如下：给定 n 个目标（生产目标）和 m 种用于实现这些目标的方法（固定的资源、设备等）。假设每个目标有若干方法可以实现，则目的是分配各种方法以完成各种目标，以最小化成本函数。这样的问题在各种工厂规模，以及更大规模的各种领域中出现。这类生产计划问题可以描述为广义运输模型（见下一节）。

7.1.2 运输问题的构造

生产计划和库存控制　在 7.1.1 节中描述的多阶段生产计划和库存控制问题可以构造为

$$
\begin{aligned}
\min\quad & 4.0x_{11}+6.0x_{21}+5.5x_{12}+7.5x_{22}+4.0x_{32}+6.0x_{42}\\
& +7.0x_{13}+9.0x_{23}+5.5x_{33}+7.5x_{43}+4.0x_{53}\\
& +6.0x_{63}+8.5x_{14}+10.5x_{24}+7.0x_{34}+9.0x_{44}\\
& +5.5x_{54}+7.5x_{64}+4.0x_{74}+6.0x_{84}\\
\text{s.t.}\quad & x_{11}+x_{12}+x_{13}+x_{14}+x_{15}=50\\
& x_{21}+x_{22}+x_{23}+x_{24}+x_{25}=20\\
& x_{32}+x_{33}+x_{34}+x_{35}=50\\
& x_{42}+x_{43}+x_{44}+x_{45}=20\\
& x_{53}+x_{54}+x_{55}=50
\end{aligned}
$$

$$x_{63}+x_{64}+x_{65}=20$$
$$x_{74}+x_{75}=50$$
$$x_{84}+x_{85}=20$$
$$x_{11}+x_{21}=40$$
$$x_{12}+x_{22}+x_{32}+x_{42}=60$$
$$x_{13}+x_{23}+x_{33}+x_{43}+x_{53}+x_{63}=75$$
$$x_{14}+x_{24}+x_{34}+x_{44}+x_{54}+x_{64}+x_{74}+x_{84}=60$$
$$x_{15}+x_{25}+x_{35}+x_{45}+x_{55}+x_{65}+x_{75}+x_{85}=45$$
$$x_{ij}\geqslant 0, \quad i=1,2,\cdots,8, \quad j=1,2,\cdots,5$$

其中，$x_{i5}(i=1,2,\cdots,8)$是对应虚拟终点 5 的变量。

一般地，具有 n 个给定周期的这类问题可以构造为下面的运输模型：

$$\min \quad \sum_{j=1}^{n}\sum_{i=1}^{2j} c_{ij}x_{ij} \tag{7.1}$$

$$\text{s.t.} \quad \sum_{j=k}^{n+1} x_{ij} = a_i, \quad i = 2k-1, \quad k = 1,2,\cdots,n \tag{7.2}$$

$$\sum_{j=k}^{n+1} x_{ij} = a_i, \quad i = 2k, \quad k = 1,2,\cdots,n \tag{7.3}$$

$$\sum_{i=1}^{2j} x_{ij} = b_j, \quad j = 1,2,\cdots,n \tag{7.4}$$

$$x_{ij} \geqslant 0, \quad i = 1,2,\cdots,2n, \quad j = 1,2,\cdots,n+1 \tag{7.5}$$

虚拟终点为 $n+1$，其中 a_i 是在第 i 个正常或加班周期的生产能力，b_j 是在第 j 个正常或加班周期制造的生产总量。这类问题可视为**平衡运输问题**（balanced transportation problem），有以下的平衡条件：

$$\sum_{i=1}^{2n} a_i = \sum_{j=1}^{n+1} b_j$$

线性运输问题(linear transportation problem)：运输问题包含两组限制：一组与源点相关的 m 个限制，节点用 S 表示；一组是与终点相关的 n 个限制，终点用 D 表示。这样，在问题中将有 mn 个变量，每个变量对应一条从源点到终点的弧。因此，下面的图是一个有向图，它的特征是二分图(bipartite graph)；即，这个网络模型的节点可以分成两部分：S 和 D，如图 7.1 所示。运输问题的目的就是找到成本最小的运输模式。具有 m 个工厂和 n 个仓库的运输问题可以用下面的公式表示：

$$\min \quad z(\boldsymbol{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{7.6}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} \leqslant a_i, \quad i = 1,2,\cdots,m \tag{7.7}$$

$$\sum_{i=1}^{m} x_{ij} \geqslant b_j, \qquad j = 1,2,\cdots,n \tag{7.8}$$

$$x_{ij} \geqslant 0, \qquad \forall i,j \tag{7.9}$$

其中，x_{ij}是从源点i到终点j的未知运输量，c_{ij}是从源点i到终点j的单位运输成本。a_i是在源点i的可供应量，b_j是在终点的需求量。

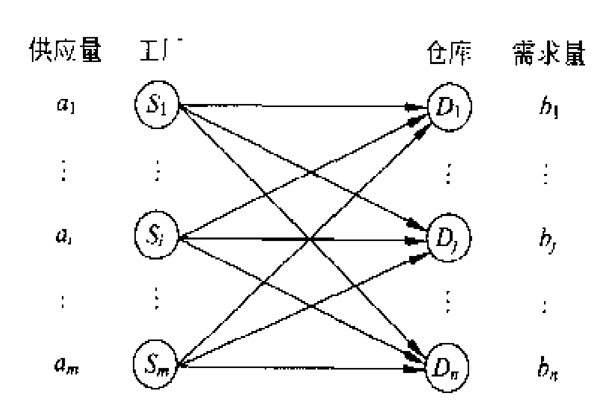


图 7.1 运输问题的网络

运输问题的解具有一些共同的特征：

1. 有 $m+n-1$ 个基变量，每个变量对应运输表中的一格。
2. 基必须是一棵运输树，即，在运输表的每一行和每一列中至少存在一个基本单位。
3. 基不能包含圈。

图 7.2 中的一个基本解解释了运输问题的特征。

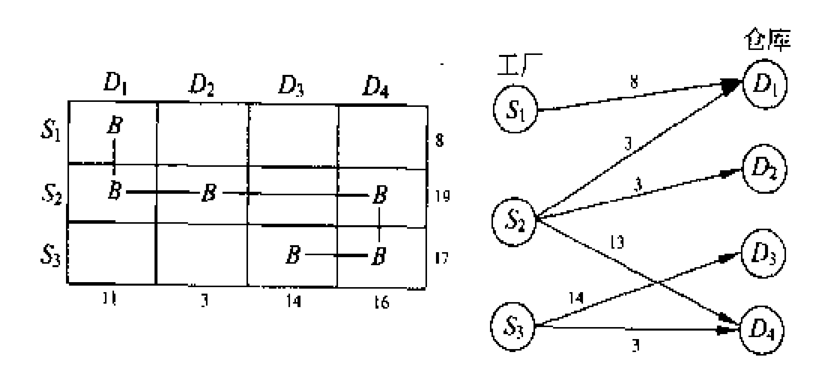


图 7.2 运输表中的基本解和运输图

由于运输规划问题的复杂性，可以考虑下面的非线性目标函数。

- 非线性函数 1：

$$f_{ij}(\boldsymbol{x}) = c_{ij} x_{ij}^2$$

- 非线性函数 2：

$$f_{ij}(\boldsymbol{x}) = c_{ij} \sqrt{x_{ij}}$$

- 非线性函数3：

$$f_{ij}(\boldsymbol{x})=\begin{cases}c_{ij}\left(\dfrac{x_{ij}}{S}\right), & \text{if } 0\leqslant x_{ij}<S\\ c_{ij}, & \text{if } S<x_{ij}\leqslant 2S\\ c_{ij}\left(1+\dfrac{x_{ij}-2S}{S}\right), & \text{if } 2S<x_{ij}\end{cases}$$

其中，S 是典型 x 值的阶。

- 非线性函数4：

$$f_{ij}(\boldsymbol{x})=c_{ij}x_{ij}\left[\sin\left(x_{ij}\frac{5\pi}{4S}\right)+1\right]$$

对于多商品的运输问题，我们也可以容易地构造一些模型，解决原问题的子问题。

广义运输问题 广义运输问题（generalized transportation problem）是一个扩展的运输问题。这类问题通常可以描述为下面的形式：

$$\min \quad z(\boldsymbol{x})=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij} \tag{7.10}$$

$$\text{s.t.} \quad \sum_{j=1}^{n}x_{ij}=a_i, \quad i=1,2,\cdots,m \tag{7.11}$$

$$\sum_{i=1}^{m}p_{ij}x_{ij}=b_j, \quad j=1,2,\cdots,n \tag{7.12}$$

$$x_{ij}\geqslant 0, \quad \forall i,j \tag{7.13}$$

其中，p_{ij} 是运输的数量单位。

容量限制的运输问题 容量限制的运输问题（capacitated transportation problem）是一类扩展的运输问题。不同于其他运输问题，这类问题在每条路线上有运输容量，其数学模型可以表示如下：

$$\min \quad z(\boldsymbol{x})=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij} \tag{7.14}$$

$$\text{s.t.} \quad \sum_{j=1}^{n}x_{ij}=a_i, \quad i=1,2,\cdots,m \tag{7.15}$$

$$\sum_{i=1}^{m}p_{ij}x_{ij}=b_j, \quad j=1,2,\cdots,n \tag{7.16}$$

$$0\leqslant x_{ij}\leqslant u_{ij}, \quad \forall i,j \tag{7.17}$$

其中，u_{ij} 是运输容量的上界。

7.2 基于生成树的方法

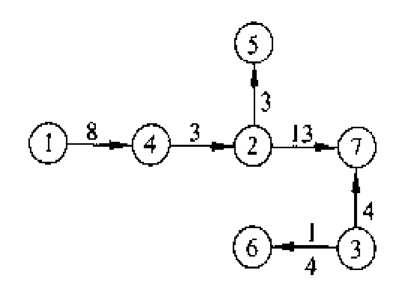
作为一种特殊类型的网络问题，运输问题有一种特殊的数据结构，在它们的解中表现为运输图。结合运输问题的数据结构，基于生成树的遗传算法（spanning tree-based

genetic algorithm)是由 Gen 和 Li 提出的[234,412]。该遗传算法使用基于生成树的 Prüfer 数编码,它可以表示所有可能的树。使用 Prüfer 数表示法,对于运输问题中的每个染色体仅需要 $m+n-2$ 个存储单元。因为运输树是一种特殊类型的树,所以 Prüfer 数可能对应不可行的解。基于这点,Gen 和 Li 设计了检验染色体可行性的准则。基于生成树的遗传算法能找到在解空间中运输问题的最优或近似最优解。

7.2.1 树的表示

定义工厂 $1,2,\cdots,m$ 为集合 $S=\{1,2,\cdots,m\}$ 的元素,仓库 $1,2,\cdots,n$ 为集合 $D=\{m+1,m+2,\cdots,m+n\}$ 的元素。显然,运输问题在它的运输树中有 $m+n$ 个节点和 $m\times n$条边。

图 7.2 中的运输图可以表示为如图 7.3 所示的一棵生成树。对于一个拥有 p 个节点的完备图,有 p^{p-2}棵标号不同的树。这意味着我们能仅仅用 $p-2$ 个数字的排列就可以惟一表示 p 个节点的树,其中每个数字是 1 至 p 中的一个数,而任何树至少有 2 片叶子。



Prüfer 数: $P(T)=[4\ 2\ 2\ 7\ 3]$

图 7.3 生成树和它的 Prüfer 数

Prüfer 数作为一种树的节点编码方法,可用于编码运输树。树的 Prüfer 数构造的过程如下:

过程 将一棵树转化为一个 Prüfer 数

第 1 步:令 i 是树 T 中数字最小的叶子节点,j 为 i 的关联节点。然而,则 j 是 Prüfer 数 $P(T)$最右边的数字。$P(T)$通过从右边添加数字构建。这样就构建了 $P(T)$,$P(T)$从左往右解读。

第 2 步:进一步考虑,移去 i 和边(i,j)。这样,节点 i 就根本不用再考虑,如果 i 是 j 的惟一后继节点,则 j 成为叶子。

第 3 步:如果只剩下两个节点可考虑,$P(T)$形成,停止;否则,返回第 1 步。

图 7.3 中解释的运输树图可以转化为如图 7.4 所示的相应的 Prüfer 数。

在运输问题中的每个节点都有它的供应量或需求量,作为限制条件。因此,构建一棵运输树时,节点的限制必须考虑。利用 Prüfer 数,通过下面的过程可以产生惟一的一棵

运输树：

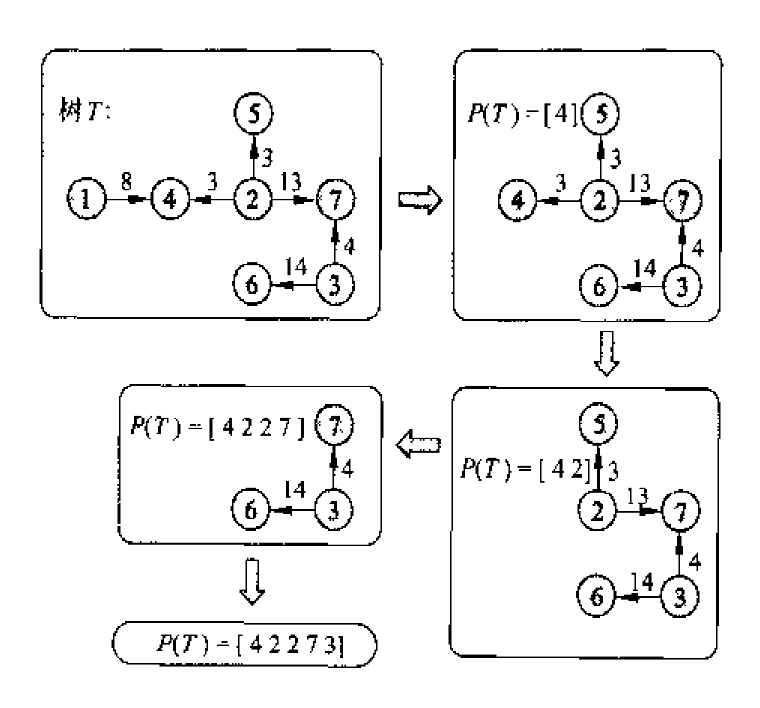


图 7.4 编码过程

过程 将 Prüfer 数转化为运输树

第1步：令 $P(T)$是原始 Prüfer 数，$\bar{P}(T)$是所有不在 $P(T)$中，可考虑的节点集合。

第2步：重复步骤(2.1)到(2.5)直至在 $P(T)$左边没有数字

(2.1) 令 i 是 $\bar{P}(T)$中数字最小的可考虑的节点，j 是 $P(T)$最左边的数字。

(2.2) 如果 i 和 j 不同在 S 或 D 中，则添加边(i,j)到树 T。否则，从 $P(T)$中选择下一个数字 k，它与 i 不在同一个集合中，用 k 交换 j，添加边(i,k)到树 T。

(2.3) 从 $P(T)$中移去 j(或 k)，从 $\bar{P}(T)$中移去 i。如果节点 j(或 k)不再在 $P(T)$余下部分的任何位置出现，则将其放入 $\bar{P}(T)$。令 i 为不可考虑的节点。

(2.4) 指定边(i,j)(或(i,k))的运输量为 $x_{ij}=\min\{a_i,b_j\}$($x_{ik}=\min\{a_i,b_k\}$)，其中$i\in S,j,k\in D$。

(2.5) 更新可用的运输量 $a_i=a_i-x_{ij}$，$b_j=b_j-x_{ij}$(或 $b_k=b_k-x_{ik}$)。

第3步：如果在 $P(T)$中不存在数字，则在 $\bar{P}(T)$还存在两个可考虑的节点 i 和 j。添加边(i,j)到树 T，形成 $m+n-1$ 条边的树。

第4步：如果没有可用的运输量可分配，则停止；否则，保留供应点 r 和需求点 s。添加边(r,s)到树，指定边的可用运输量为 $x_{rs}=a_r=b_s$。如果此时存在圈，则移去流量为0的边。从而形成具有 $m+n-1$ 条边的树。

图 7.5 解释了从 Prüfer 数到运输树的过程。

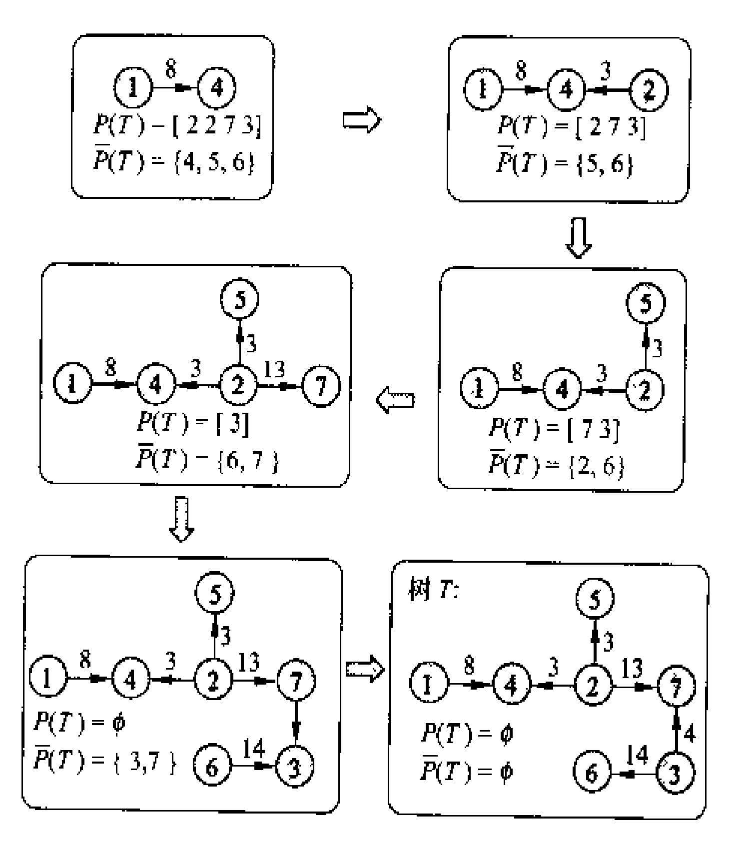


图 7.5 解码过程

7.2.2 初始化

染色体(Prüfer 数)的初始化由$[1,m+n]$中随机产生的 $m+n-2$ 个数字开始执行。然而,这有可能产生不可行的染色体,即不适合产生运输树。Prüfer 数编码不仅仅能等价惟一的代表所有可能的生成树,而且明确地包含了关于节点度的信息。也就是说,任何度为 d 的节点在编码中会出现 $d-1$ 次。这意味着当节点在 Prüfer 数中出现 d 次时,该节点与其他 $d+1$ 个节点相连。

Gen 和 Li 设计了一用于检验染色体可行性的准则。令 $\bar{P}(T)$为不包含在 Prüfer 数 $P(T)$中的节点集合。L_S 为由 Prüfer 数 $P(T)$定义的生成树中与工厂节点($|S\cap P(T)|$)相关的边的总数。L_S 为同一生成树中与仓库节点($|D\cap P(T)|$)相关的边的总数。$\bar{L}_S$ 是包含在 $\bar{P}(T)$中的工厂节点数,即,$\bar{L}_S=|S\cap\bar{P}(T)|$。$\bar{L}_D$ 是包含在 $\bar{P}(T)$中的仓库节点数,即,$\bar{L}_D=|D\cap\bar{P}(T)|$。我们有以下的判别准则:

染色体的可行性:如果 $L_S+\bar{L}_S=L_D+\bar{L}_D$,则 $P(T)$是可行的;否则 $P(T)$不可行。

例如,Prüfer 数 $P(T)$是$[4\ 2\ 2\ 7\ 3]$,$\bar{P}(T)$是$\{1,5,6\}$,见图 7.5。L_S 是与节点 2 和 3 相连的边的总数。节点 2 和 3 包含在 $P(T)$的集合 S 中。与节点 2 相连的边有 2+1 条,与节点 3 相连的边有 1+1 条,故 $L_S=5$。同样,L_D 是与包含在集合 D 中的节点 4 和 7 相连的边的总数,$L_D=4$。$\bar{L}_S$ 是节点 1 在 $\bar{P}(T)$中出现的次数,因此 $\bar{L}_S=1$。$\bar{L}_D$ 是节点 5 和 6 出现的次数,因此 $\bar{L}_D=2$。因为 $L_S+\bar{L}_S=6$,$L_D+\bar{L}_D=6$,所以这个 Prüfer 数满足可行性准则。

7.2.3 遗传运算

杂交 如图7.6所示使用了单分割点杂交运算。为了避免对那些在杂交运算后生产的不可行染色体的不必要的解码,检验过程被嵌入杂交运算,以保证产生可行的子代,也就是说,通过这个检验过程,子代永远与运输树相一致。

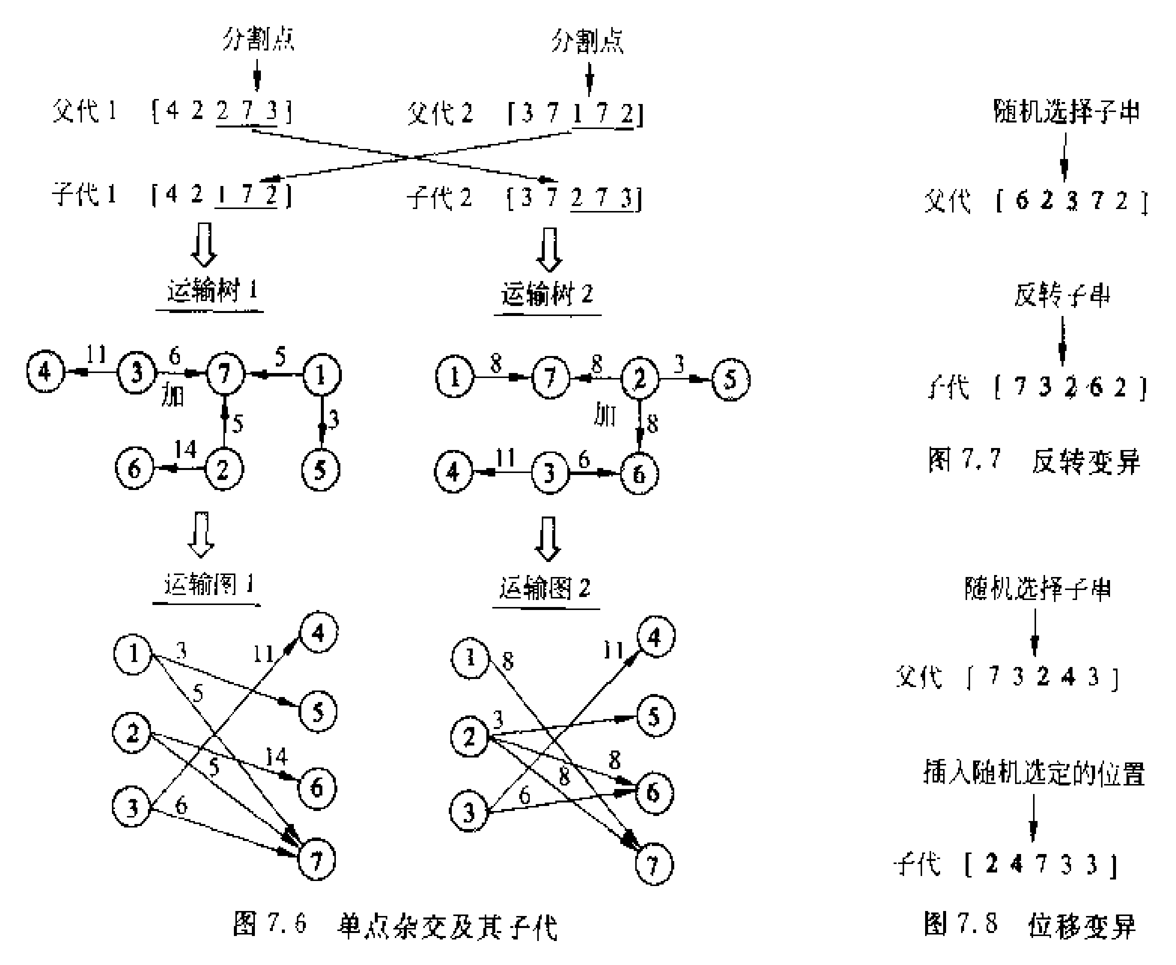


图7.6 单点杂交及其子代

图7.7 反转变异

图7.8 位移变异

变异 使用反转变异和位移变异。利用反转变异,随机地选择染色体中的两个位置,然后如图7.7所示将两个位置间的子串反转。利用位移变异,随机选择位移子串,如图7.8所示插入随机选定的位置。如果父代是可行的,则这两种变异永远产生可行的染色体,因为在运算后,准则 $L_S+\bar{L}_S=L_D+\bar{L}_D$ 是不变的。

7.2.4 评价与选择

评价过程由两步组成:

1. 将染色体转换为树。
2. 计算每个目标函数。

过程 评价

第1步:令 T 是树,设置其为一空树,$T=\{\varnothing\}$。令 p 是数位计数器,并置零,$p=0$。设置

每个目标函数值为 0。

第 2 步：根据 Prüfer 数 $P(T)$定义 $\bar{P}(T)$。

第 3 步：只要 $p \leqslant m+n-2$ 重复步骤(3.1)至(3.6)

(3.1) 选择 $P(T)$最左边的数字 i。从 $\bar{P}(T)$中选择数字最小的可考虑节点，标记为 j。

(3.2) 如果 $i,j \in S$ 或 $i,j \in D$，则从不存在 j 的集合 $P(T)$中选择下一个数字。标记为 k，i 和 k 交换，令 $i=k$。添加边(i,j)到树 T。

(3.3) 从 $P(T)$中移去 i，从 $\bar{P}(T)$中移去 j。如果 j 不再出现 $P(T)$余下部分的任何位置，则将其加入 $\bar{P}(T)$。令 j 为不可考虑的节点。

(3.4) 指定边(i,j)的流量为 $x_{ij}=\min\{a_i,b_j\}$，其中 $i \in S, j \in D$。更新边(i,j)的目标函数值。

(3.5) 更新可用的运输量 $a_i=a_i-x_{ij}$，$b_j=b_j-x_{ij}$。

(3.6) 令 $p=p+1$。

第 4 步：如果在 $P(T)$中不存在数字，则在 $\bar{P}(T)$还存在两个可考虑的节点 r 和 s。添加边(r,s)到树 T，形成 $m+n-1$ 条边的树。指定边(r,s)的流量为 $x_{rs}=\min\{a_r,b_s\}$，其中 $r \in S, s \in D$。更新边(r,s)的目标函数值。

第 5 步：重复步骤(5.1)和(5.2)直至在任何节点没有可用的运输量。

(5.1) 如果 $a_i>0, \forall i, b_j>0, \forall j$，则添加边$(i,j)$到树 T，$T=T \cup \{x_{ij}\}$，并指定流量 $x_{ij}=\min\{a_i,b_j\}$，其中 $i \in S, j \in D$。

(5.2) 更新可用的运输量 $a_i=a_i-x_{ij}$，$b_j=b_j-x_{ij}$。更新边(i,j)的目标函数值。

第 6 步：如果存在圈，则确定流量为 0 的边，并移去。

选择过程结合了最优性轮盘赌选择和$(\mu+\lambda)$-选择。这种混合选择策略从 μ 个父代和 λ 个子代中选择 μ 个最佳染色体。如果不存在可行的 μ 个不同染色体，则使用最优性轮盘赌选择填充种群池的空缺。确定性的$(\mu+\lambda)$-选择能迫使最佳的染色体进入下一世代中，避免进化过程的过早收敛[414]。

7.2.5 整个算法过程

令 $P(t)$是染色体种群，$C(t)$是在当前世代 t 中产生的染色体。整个算法过程如下：

Procedure: **st-GA/TP**

Begin

　$t \leftarrow 0$;

　初始化 $P(t)$;

　基于生成树评价 $P(t)$;

　while 终止条件不满足 **do**

　begin

　　重组 $P(t)$ 以产生 $C(t)$;

　　基于生成树评价 $C(t)$;

```
        从 P(t) 和 C(t)中选择 P(t+1);
        t ← t+1;
    end
end
```

7.3 多目标运输问题

在经济决策中，往往需要多个目标的表示。一般来说，我们可以考虑包含所有目标的效用体系的最大化。然而，模型的使用经常需要有关结构性的或中间目标的具体规范。或许，需要设计目标的相对重要性，或关于单个或多个目标所要求(允许)达到的最小(最大)化水平规范，且要求这类规范最小化。

在一般运输问题中，目标是最小化总运输成本。使用这种方法的基本假设是：管理主要与成本最小化相联。然而，这个假设并不总是有效的。Gen，Li 和 Ida[226,233,716]认为，在运输问题中可能有多个目标，例如：运输计划合同的履行，联合合同的履行，提供各类工厂和运输队稳定的雇工水平，许多工厂间工作的平衡，运输风险最小化和成本的最小化。

一般地说，一个多目标线性规划问题可以写成：

$$\max \quad z_1(\boldsymbol{x}) = c^1 \boldsymbol{x} \tag{7.18}$$

$$\max \quad z_2(\boldsymbol{x}) = c^2 \boldsymbol{x} \tag{7.19}$$

$$\vdots \tag{7.20}$$

$$\max \quad z_q(\boldsymbol{x}) = c^q \boldsymbol{x} \tag{7.21}$$

$$\text{s. t.} \quad \boldsymbol{x} \in S \tag{7.22}$$

其中，q 是目标数，c^i 是第 i 个目标函数的向量的系数(梯度)，S 为可行域，$z_i(\boldsymbol{x})$是第 i 个目标函数值(目标值)。max 表示目标是同时求所有目标的最大化。

因为多目标问题很少存在同时使得所有目标最大化的解，所以我们处于这样的情况：最大程度地最大化每个目标。许多研究者对多目标运输问题(multiple-objective transportation problem)感兴趣，并提出了许多解决它的方法。

7.3.1 问题的描述

在运输问题中，例如：最小化运输成本、最小化对于优先顾客的平均装运时间、最大化生产量、最小化燃料消耗等多目标，在实际情况下往往是必要的。具有 m 个工厂和 n 个仓库的传统多目标运输问题可以描述如下：

$$\min \quad z_q = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^q x_{ij}, \quad q = 1,2,\cdots,Q \tag{7.23}$$

$$\text{s. t.} \quad \sum_{j=1}^{n} x_{ij} \leqslant a_i, \qquad i = 1,2,\cdots,m \tag{7.24}$$

$$\sum_{i=1}^{m} x_{ij} \geqslant b_j, \qquad j = 1,2,\cdots,n \tag{7.25}$$

$$x_{ij} \geqslant 0, \qquad \forall\, i,j \tag{7.26}$$

其中,q 表示第 q 个目标函数。

7.3.2 多目标运输问题的基于生成树的遗传算法

Pareto 最优解通常被描述成多目标规划问题的解。因此,在遗传算法的执行中,需要增加处理 Pareto 最优解的模块[233]。它包含两步:

1. 利用目标函数评价染色体。
2. 基于评价值选择 Pareto 解。

设 $E(t)$ 是到当前迭代世代 t 产生的 Pareto 解集,下面给出求 Pareto 解的过程:

过程 Pareto 解集

第 1 步:设置迭代标志 $k=1, E(t)=\{\varnothing\}$。

第 2 步:如果 $k>i_size$,则停止;否则,执行第 3 步。

第 3 步:评价染色体 T_k,得到解向量 $Z_k=[z_1(T_k)z_2(T_k)\cdots z_Q(T_k)]$。

第 4 步:将它与 E 中所有 Pareto 解进行比较。

(4.1) 如果任何 Pareto 解优于它,执行第 5 步。

(4.2) 如果它优于部分 Pareto 解,则将其增加入 E,并将 E 中被其优超的解删除。

(4.3) 如果它是一个新 Pareto 解,不优于任何存在解,则简单地将其增加入 E。

第 5 步:设置 $k=k+1$,返回第 2 步。

加权和的方法被用于构建适应值函数。多目标函数 $z_q, q=1,2,\cdots,Q$,可以归结为一个整体的目标函数。

适应值函数的处理:

1. 对应于每个目标函数的最小值 $z_q^{\min}$(或最大值)$z_q^{\max}$ 的解,并将它们与先前保存下来的解进行比较,再次选择最佳解,并给予保存。

$$\left.\begin{aligned} z_q^{\min(t)} &= \min_k\{z_q^{\min(t-1)}, z_q^{(t)}(T_k) \mid k=1,2,\cdots,i_size\}, \\ z_q^{\max(t)} &= \max_k\{z_q^{\max(t-1)}, z_q^{(t)}(T_k) \mid k=1,2,\cdots,i_size\}, \end{aligned}\right. \quad q=1,2,\cdots,Q$$

其中,$z_q^{\max(t)}[z_q^{\min(t)}]$是在 t 世代目标函数的最大(最小)值,i_size 是遗传运算后产生的子代。

2. 求解下面的等式,得到适应值函数的权重:

$$\delta_q = z_q^{\max(t)} - z_q^{\min(t)}, \quad \beta_q = \frac{\delta_q}{\sum_{q=1}^{Q}\delta_q}, \quad q=1,2,\cdots,Q$$

3. 计算每个染色体的适应值:

$$\mathrm{eval}(T_k) = \sum_{q=1}^{Q}\beta_q z_q(T_k), \quad k=1,2,\cdots,i_size$$

整个算法过程　令 $P(t)$ 是当前世代 t 时的染色体种群，$C(t)$ 是当前世代 t 产生的染色体，$E(t)$ 是到当前世代 t 为止产生的 Pareto 解集，则整个算法过程归结如下：

```
procedure: st-GA/mTP
begin
    t ← 0;
    初始化 P(t);
    应用适应值函数评价 P(t);
    确定 Pareto 解集 E(t);
    while 终止条件不满足 do
    begin
        重组 P(t)以产生 C(t);
        更新 Pareto 解集 E(t);
        应用适应值函数评价 C(t);
        从 P(t)和 C(t)中选择 P(t+1);
        t ← t+1;
    end
end
```

混合遗传算法　为了提高基于生成树的遗传算法的有效性，Gen 和 Li 提出了混合遗传算法方法，在基于生成树的遗传算法中结合简约成本的一种局部改进过程[232]。

简约成本在运输问题的简单模型中最初被用于检验解的最优性。在单一目标的情况下，如果对于所有的非基格(i,j)不存在简约成本 $\bar{c}_{ij}$，则当前的解是最优解。受这种思想的激发，设计了局部改进过程。在多目标情况下，对于每个非基格（变量 x_{ij}），存在 Q 个简约成本 $\bar{c}_{ij}^q$，$q=1,2,\cdots,Q$。令 S 为工厂节点集，D 是仓库节点集。局部改进过程如下：

第 1 步：为每个非基格(i,j)计算简约成本 $\bar{c}_{ij}^q$，$q=1,2,\cdots,Q$，其中，$i\in S,j\in D$。

第 2 步：选择 Q 个格(k,l)，$q=1,2,\cdots,Q$，其中：

$$c_{kl}^q=\max\{\bar{c}_{ij}^q\mid\bar{c}_{ij}^q>0,\text{对于所以非基弧}(i,j)\},q=1,2,\cdots,Q$$

第 3 步：分别将 Q 条弧加入当前树中，产生 Q 棵不完整的树 U_q，$q=1,2,\cdots,Q$。

第 4 步：在不完整的树 U_q 上生产的圈中，$q=1,2,\cdots,Q$，寻找最小流 Δ_q，$q=1,2,\cdots,Q$。

第 5 步：在不完整树 U_q 上，$q=1,2,\cdots,Q$，在圈的周围沿同一方向产生一个容量为 $+\Delta_q$ 的流；沿反方向产生一个容量为 $-\Delta_q$ 的流。移去流量减为 0 的弧，这就产生了新树 T_q，$q=1,2,\cdots,Q$。

第 6 步：计算 T_q，$q=1,2,\cdots,Q$ 的目标函数值，比较每个目标函数值的增量。

第 7 步：从 T_q，$q=1,2,\cdots,Q$ 中选择一棵有一个目标函数值异常增值的树。

运算后，我们总能找到一个比原先的运输树解更好的解。

整个混合遗传算法的计算过程如下：

```
procedure: st-GA/mTP
begin
    t ← 0;
    初始化 P(t);
    应用适应值函数评价 P(t);
    确定 Pareto 解集 E(t);
    while 终止条件不满足 do
    begin
        重组 P(t)以产生 C(t);
        应用适应值函数评价 C(t);
        更新 Pareto 解集 E(t);
        从 P(t)和 C(t)中选择 P(t+1);
        用局部改进方法更新 P(t);
        t ← t+1;
    end
end
```

7.3.3 数例

提出的基于生成树的遗传算法(st-GA)和混合遗传算法(st-HGA)方法用C语言实现,并在装有 EWS-UX/V release 4.0 UNIX OS 的 NEC EWS 4800 上运行。首先,为了确认关于运输问题的基于生成树的遗传算法的有效性,在计算研究中使用了5个随机产生的数字实例。第一个例子来自文献[14]。得到的解仅考虑惟一的目标函数。在随机产生的例中,供应量、需求量和运输成本都是随机生产的,符合[100,50]和[10,50]上的均匀分布。

对每个例子,在最好的参数设置下进行模拟运算。表 7.2 给出了每个问题的基于生成树的遗传算法(st-GA)和基于矩阵的遗传算法(m-GA)在 10 次运行中得到的平均计算结果。从表 7.2 中我们知道 st-GA 比 m-GA 消耗更少的时间找到最优解。这意味解决运输问题,st-GA 比 m-GA 更适合。

表 7.2 对于运输问题 st-GA 和 m-GA 方法的平均结果

问题大小	参数		m-GA		st-GA	
$m\times n$	种群大小	最大世代	%	ACT/min	%	ACT/min
3×4	10	200	100	0.017	100	0.017
4×5	20	50	100	0.212	100	0.146
5×6	20	500	100	0.313	100	0.166
6×7	100	1000	100	17.304	100	6.893
8×9	200	2000	100	74.536	100	14.135

注: %为求得最优解次数占运行次数的百分比; ACT 为平均计算时间。

在两个数字实例中，我们测试了解决双目标运输问题时 st-GA 和 m-GA 的性能。第一个例子（来自文献[14]）中有 3 家工厂、4 个仓库，供应量、需求量、成本见表 7.3。

表 7.3 第一个双目标运输问题的供应量、需求量和成本

目的地，b_j 源节点，a_i	c_{ij}^1				c_{ij}^2				供应量
	4	5	6	7	4	5	6	7	
1	1	2	7	7	4	4	3	4	8
2	1	9	3	4	5	8	9	10	19
3	8	9	4	6	6	2	5	1	17
需求量	11	3	14	16	11	3	14	16	44

st-GA 和 m-GA 的参数设置如下：$pop_size=30$，$p_c=0.2$，$p_m=0.4$，$max_gen=1000$，$run=20$ 次。总是可以求得 Pareto 最优解，它们是：(143,265)，(156,200)，(176,175)，(186,171)，(208,167)。众所周知，Pareto 最优解可以形成 Pareto 前沿面。如图 7.9所示，两个端点是(143,265)和(208,167)。st-GA，m-GA 和 st-HGA 的平均计算时间分别为 13.50、14.21、18.16s。这说明对于小规模的问题，在 st-GA、m-GA 和 st-HGA 三种情况下，计算时间无明显差异。

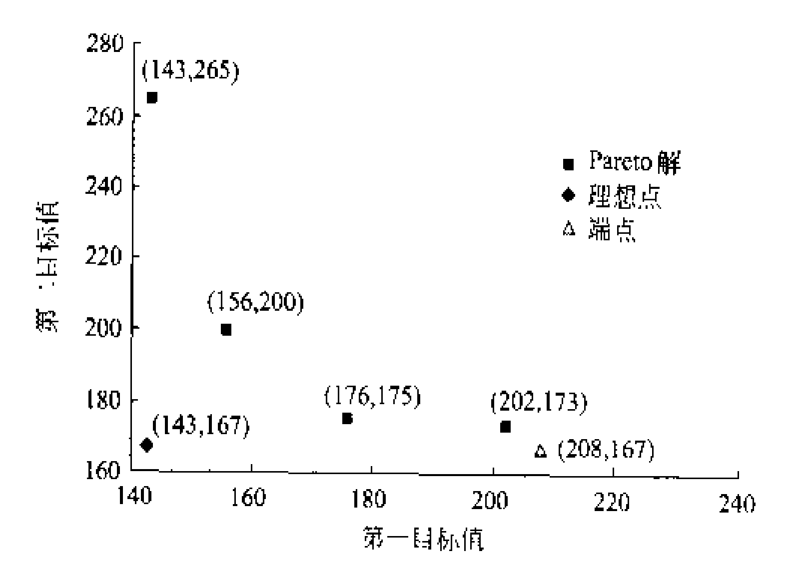


图 7.9 由 st-GA 得到的 Pareto 最优解

第二个例子中有 8 家工厂和 9 个仓库。两个目标函数的参数见表 7.4。算法的参数设置为：$pop_size=100$，$max_gen=500$，在较好的参数配置下($p_c=0.2$，$p_m=0.4$)运行 20 次。图 7.10给出了 st-GA，m-GA 和 st-HGA 的比较结果。运用 m-GA 时，参数设置为：$p_c=0.4$，$p_m=0.2$；运用 st-GA 和 st-HGA 时，参数设置为：$p_c=0.2$，$p_m=0.4$。所有的例子均运行了 20 次。从图 7.10 中，我们可以看到在 Pareto 最优性上，用 st-GA 得到的结果要优于 m-GA 的结果，因为大部分的结果和用 m-GA 得到的一样优秀。因此，我

们可以得出结论：它们比用 m-GA 得到的解更接近真实的 Pareto 前沿面。而且，在 Pareto 最优性上，用 st-HGA 得到的结果要优于 st-GA 的结果。明显地，在多目标问题上，st-HGA 比st-GA 更有效。

表 7.4 第二个双目标运输问题的供应量、需求量和成本

D_j / S_i	c_{ij}^1 9	10	11	12	13	14	15	16	17	c_{ij}^2 9	10	11	12	13	14	15	16	17	供应量 a_i
1	1	2	7	7	8	10	9	2	5	4	4	3	4	5	8	9	10	2	10
2	1	9	3	4	3	5	7	1	1	6	2	5	1	7	4	12	4	4	8
3	8	9	4	6	4	1	6	9	2	2	9	1	8	9	1	4	0	1	12
4	2	4	5	5	3	2	3	2	9	3	5	5	3	2	8	3	3	2	16
5	5	4	5	1	9	9	1	6	1	1	4	12	2	1	5	4	9	1	21
6	8	3	3	2	2	3	6	7	6	2	23	4	4	6	2	4	6	7	15
7	1	2	6	4	5	9	3	5	2	1	2	1	9	0	13	2	3	2	7
8	13	3	3	5	1	5	6	3	2	14	3	4	2	1	8	5	3	1	9
需求量 b_j	9	7	15	10	13	16	7	10	11	9	7	15	10	13	16	7	10	11	98

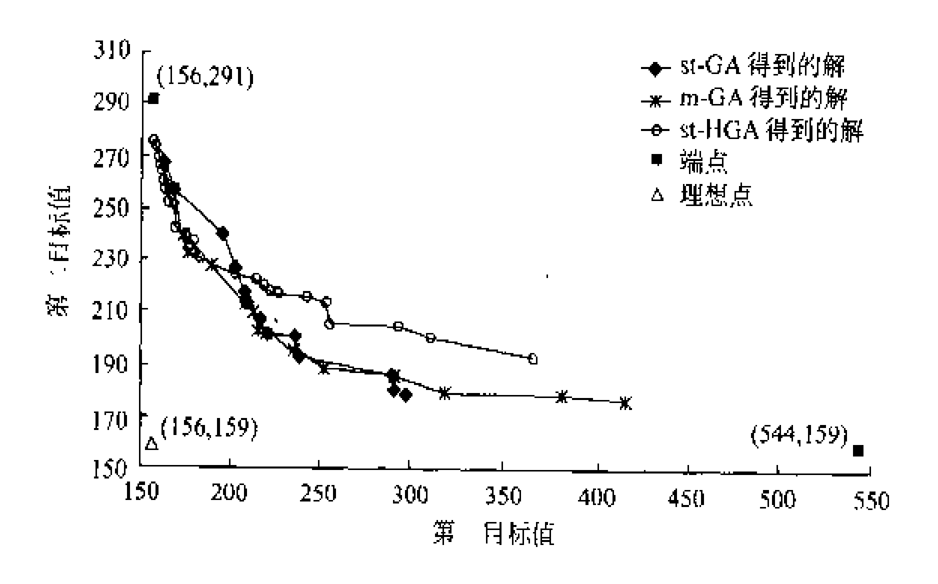


图 7.10 m-GA，st GA 和 st-HGA 的 Pareto 最优解的比较

此外，m-GA，st-GA 和 st-HGA 的平均计算时间分别为 13.387，1.159，3.142min。显然，在减少计算时间方面，st-GA 比 m-GA 和 st-HGA 更有效，因为基于生成树的编码，每个染色体仅需要 $m+n-2$ 个存储单元，而基于矩阵的编码为代表每条染色体需要 $m\times n$ 个储存单元。由于 st-HGA 是局部改进与 st-GA 的混合形式，所以求解过程中，它比要 st-GA 花费更多的计算时间。增加存储单元的结果是，基于矩阵的编码比基于生成树的编码遗传算法花费更多的时间。因此，对于大规模问题，基于生成树编码的遗传算法比基于矩阵编码和混合遗传算法更节省时间(见表 7.5)。

表 7.5　m-GA,st-GA 和 st-HGA 方法的比较

m	n	解的存储单元大小		ACT/min			参　数	
		m-GA	st-GA(st-HGA)	m-GA	st-GA	st-HGA	种群大小	最大世代数
3	4	12	5	0.236	0.225	0.302	30	1000
8	9	72	15	13.387	1.159	3.142	100	500
10	15	100	23	41.957	15.363	27.060	150	500

注：m 为工厂数量；n 为仓库数量；m-GA，基于矩阵的遗传算法；st-GA，基于生成树的遗传算法；st-HGA，基于生成树的混合遗传算法。

为了验证对于多目标问题基于生成树的遗传算法的有效性和效率，Gen 和 Li 进行了数值实验，并将它与基于矩阵的遗传算法(m-GA)、基于生成树的遗传算法(st-GA)和基于生成树的混合遗传算法(st-HGA)进行了比较。对于小规模问题，结果没有明显的差异。对于大规模问题，基于生成树的遗传算法得到 Pareto 最优解所花费的 CPU 时间比基于矩阵的遗传算法要更少，大多数解优于基于矩阵的遗传算法得到的解。因此，从 Pareto 最优性看，基于生成树的遗传算法比基于矩阵的遗传算法更有效。

最近，Jiménez 和 Verdegay 提出了用于求解区间系数的单一运输问题的进化算法[723]。

7.4　固定费用运输问题

固定费用运输问题(fixed-charged transportation problem)(fcTP)是运输问题的扩展。许多实际运输和分配问题，如具有固定物流费用的最小费用网络流(转运)问题，可以用公式表示为固定费用运输问题。又如运输问题中，在给定的工厂和仓库之间每次运输存在一个固定成本，一家工厂或仓库需要固定的投资。fcTP 考虑了这些固定费用，因此，运输问题是所有路径的固定费用成为零的 fcTP 问题。fcTP 更难以处理，因为它将引起目标函数的不连续性。

7.4.1　数学模型

在固定费用运输问题中，当选择活动的最佳过程时，两类费用要同时考虑：(1)与活动水平成比例的可变成本；(2)固定成本。fcTP 为同类商品从工厂到仓库的运输，确定成本最小的运输计划。这需要有关于每个工厂的供应水平、每个仓库的需求量、从每个工厂到每个仓库的运输成本和固定成本的具体说明。目标是在满足每个仓库的需求时，分配在每家工厂的可用供应量，以优化目标函数。通常的目标函数是最小化调配的总可变成本和固定成本。

m 家工厂和 n 个仓库的固定费用运输问题可以用公式表述如下：

$$\min \quad f(\boldsymbol{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n}[f_{ij}(\boldsymbol{x}) + d_{ij}g_{ij}(\boldsymbol{x})] \tag{7.27}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} \leqslant a_i, \quad i=1,2,\cdots,m \tag{7.28}$$

$$\sum_{i=1}^{m} x_{ij} \geqslant b_j, \quad j=1,2,\cdots,n \tag{7.29}$$

$$x_{ij} \geqslant 0, \qquad \forall i,j \tag{7.30}$$

$$g_{ij}(\boldsymbol{x}) = \begin{cases} 1, & \text{如果 } x_{ij} > 0 \\ 0, & \text{否则} \end{cases} \tag{7.31}$$

其中，$\boldsymbol{x} = x_{ij}$是在路径(i,j)上运送的未知运输量，d_{ij}是与路径(i,j)相关的固定成本，a_i为工厂i的可用量，b_j是仓库j的需求量。$f_{ij}(\boldsymbol{x})$可看作原运输问题的目标函数，从工厂i到仓库j的总单位运输成本。如果它是线性的，则目标函数中的$f_{ij}(\boldsymbol{x})=c_{ij}x_{ij}$是成本函数。

7.4.2 fcTP 问题的难点

fcTP 问题的难点主要在于参数m和n的大小。然而，问题结构的其他属性使得一些实例比同样规模的其他实例更难以处理。Kennington 和 Unger[354]报道了实例难度与最优解的 F/C 比例(固定成本与可变成本之比)之间的强对应关系。F/C 比例界于 3 到 10 的问题更难。不幸的是，我们无法事先计算这个因子。因此，Palekar, Karwan 和 Zionts[498]提出了 F/C 比例可由难度比例$\bar{d}(m+n-1)/cD$近似地代替，其中D是所有需求的总和，c和$\bar{d}$分别是平均可变成本和平均固定成本。试验结果表明最难的实例其比例界于 30 和 60 之间[498]。

7.4.3 fcTP 的求解方法

针对固定费用运输问题，人们提出了许多求解的方法，包括精确解算法和启发式方法。由于精确解算法需要大量的计算时间，所以当问题的复杂性超过一定水平时，它就不是很有效了。因此，提出并开发了许多启发式方法。Hirsch 和 Dantzig[300]首先研究了固定费用问题，他们证明了最优解在问题约束条件构成的一个极值点上。

固定费用运输问题经常作为混合整数网络规划问题被构造并求解。任何混合整数规划的求解方法都可以用于求解 fcTP，例如，分支定界法或割平面法。由于它们没有利用 fcTP 的特殊网络结构，所以这些方法一般来说是低效且计算费时的[266]。

近来，对于固定费用运输问题，Gottlieb 和 Paulmann[266]提出了基于矩阵排列表示的遗传算法。Sun 等[592]提出了禁忌搜索的启发式方法。

既然固定费用运输问题是运输问题的扩展，则它有着与生成树同样特征的网络结构。因此，基于生成树的遗传算法可应用于这类问题。计算实验显示了基于生成树遗传算法的性能。

7.4.4 遗传算法的实现

遗传子表示 作为一种树的节点编码方式，Prufer 数被用于编码运输树。为了检验每条产生的染色体(Prüfer 数)，使用了染色体的可行性准则。

遗传算子 作为遗传算子，采用了单分割点杂交。在遗传运算中使用了反转变异和位移变异。

评价和选择 固定费用运输问题仅仅在运输路径上发生的固定成本方面不同于传统的运输问题。这体现在目标函数值是通过将固定成本加入到运输树上的对应边来计算的。因此，在这个问题的评价程序中，TP 中的计算式 $f(T) \leftarrow f(T)+f_{ij}(x_{ij})$ 被包含固定成本的 $f(T) \leftarrow f(T)+f_{ij}(x_{ij})+d_{ij}$ 代替。

$(\mu+\lambda)$ 选择和最优性轮盘赌选择的混合策略被使用，这能迫使最佳的染色体进入下一世代中，避免进化过程的过早收敛。

7.4.5 数例

基于生成树的遗传算法用 C 语言实现，并在 HP 9000 Model 715/100 工作站 UNIX 操作系统下运行。使用解的质量和 CPU 时间来评价求解程序的性能。为了对所提出的遗传算法和其他遗传算法进行比较，利用 Klingman，Naoier 和 Stutz[369] 提出的生成器 NETGEN 产生了测试问题，该生成器 Barr，Glover 和 Klingman[44] 对其进行了修改。

用 $m\times n$ 表示问题的大小，三个问题的规模分别为 5×10、10×10 和 10×20。不同规模问题的总供应量(需求量)见表 7.6。不同测试问题类型的整数固定成本的范围见表 7.7。所有测试问题的可变成本是从 3 到 8 的整数。表 7.8 给出了 A 类 5×10 的问题的抽样数据。

表 7.6 不同规模的问题的总供应量(需求量)

问题规模	总供应量
5×10	5000
10×10	10 000
10×20	15 000

表 7.7 不同类型测试问题的整数固定成本的范围

问题类型	固定成本的范围	
	下限	上限
A	50	200
B	100	400
C	200	800

表 7.8 A 类 5×10 的问题的抽样数据

工厂 i	运输成本 c_{ij}										供应量 a_i
1	8	4	5	6	3	6	3	5	6	7	441
2	7	6	5	5	8	8	3	8	3	6	1267
3	7	8	4	4	3	8	6	5	6	6	1834
4	5	6	4	8	7	8	5	7	6	6	500
5	4	8	6	4	7	7	8	5	3	3	958

续表

工厂 i	运输成本 c_{ij}										供应量 a_i
1	107	181	83	137	199	165	148	85	135	109	441
2	149	107	196	132	125	200	140	125	72	82	1267
3	112	184	178	89	84	82	175	63	199	151	1834
4	146	162	169	178	132	51	59	159	112	104	500
5	165	135	91	164	88	88	105	96	96	129	958
需求量 b_j	534	488	868	572	874	315	355	391	237	366	5000

使用给定的参数和编制的算法运行 10 次：变异概率 0.4，杂交概率 0.2，最大繁殖世代数为 2000，种群大小不同。对于不同大小的问题，给定的种群大小为：5×10 的种群为 200，10×10 的种群为 300，10×20 的种群为 400。对于基于矩阵的遗传算法，遗传运算概率为：变异 0.2，杂交 0.4，这是最好的参数设置。表 7.9 比较了针对三个不同类型的测试问题，基于生成树的遗传算法(st-GA)和基于矩阵的遗传算法(m-GA)的结果。为了比较，解的质量定义如下：

$$解的质量=\frac{最优解}{得到的解}\times 100$$

对于一个给定的问题，最优解是所有 m-GA 和 st-GA 解中最好的。标记"Freq."的列给出了 10 次测试中利用遗传算法树找到最优解的问题的个数。

表 7.9 两种遗传算法的解的质量的比较

问题		m-GA				st-GA			
大小	类型	最坏	平均	最优	Freq.	最坏	平均	最优	Freq.
5×10	A	90.80	95.06	98.73	0	98.49	99.12	100.00	2
	B	88.87	92.86	94.01	0	97.78	99.03	100.00	2
	C	85.41	86.99	90.82	0	98.31	99.83	100.00	9
10×10	A	99.62	99.86	100.00	4	99.42	99.77	100.00	2
	B	99.72	99.85	100.00	3	99.61	99.74	100.00	3
10×20	A	94.85	95.93	97.62	0	98.11	99.25	100.00	2

对于表 7.9 中所示的计算结果，较之基于矩阵的遗传算法，基于生成树的遗传算法在更多的情况下找到最优解。比较 fcTP 的解的质量，st-GA 比 m-GA 具有更多的进化机会求得最优解或近似最优解。表 7.10 根据结果比较了 st-GA 和 m-GA 节省的时间。当然，如果我们增加 m-GA 的种群大小，在编码空间中可以找到最优解或近似最优解，其中固定费用是较小整数(作为 A 类)，但它将占用更多的 CPU 时间。图 7.11 显示了对于 A 型 5×10 的问题，作为进化过程函数的平均解的质量。很明显，在进化过程中，为了获得

最优解,基于生成树的编码比基于矩阵的编码具有更佳的性能。近来,Gen、Li和Ida提出了用于求解双目标固定费用运输问题的基于生成树的遗传算法[234,717]。

表 7.10 对于两类遗传算法 CPU 时间的比较

问题		解的存储单元		平均 CPU 时间	
大小	类型	m-GA	st-GA	m-GA	st-GA
5×10	A	50	13	42.192	14.110
	B	50	13	42.192	12.160
	C	50	13	41.981	10.829
10×10	A	100	18	179.355	24.217
	B	100	18	178.047	24.383
10×20	A	200	28	675.819	471.758

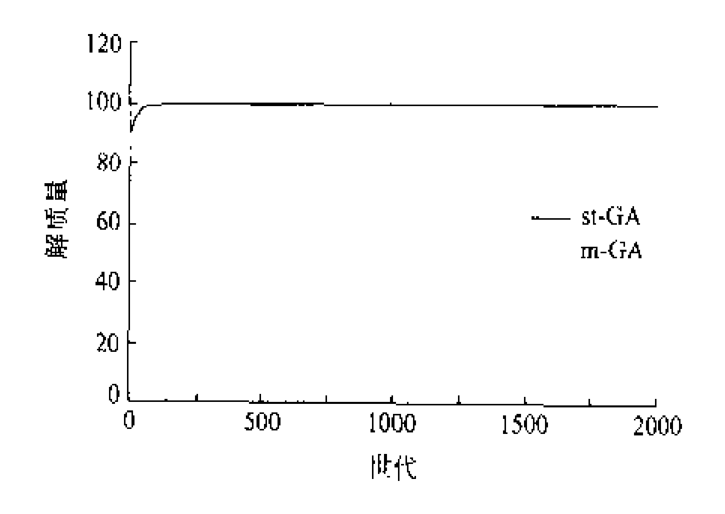


图 7.11 关于 5×10 的 fcTP 使用两类遗传算法的进化过程

7.5 容量限制的工厂选址问题

工厂选址问题 (plant location problem)涉及对工厂需求的分配。在一些情况下,某点的需求在工厂间是不能被分割的。这一点很重要。在另一些情况下,则任何有能力的工厂都可以满足需求。工厂选址模型必须反映这些不同需求的分配原则,然后必须给不同的工厂分配需求(或在某地区的总需求份额)。在许多情况下,需求被分配到最近的工厂。在另一些情况下,这样做可能并不最优。

工厂选址问题实际上是固定费用模型的变型。有 n 个接收站,供应来自 m 家潜在的工厂或仓库,以满足这些终点的需求。通常 n 要比 m 更大。为了满足需求,有必要确定每家工厂的位置和生产能力。对于每家工厂的建设存在安装(固定)成本。目标就是最小化总成本,包括固定成本和源点与终点之间的运输成本。如果对于任何 j,有 $x_{ij}>0$,则固

定成本项 d_i 出现在目标函数中；否则，不出现，其中 x_{ij} 是从源点 i 到终点 j 的运输量，这样问题将简化为普通的运输问题。很显然，对于事先确定的固定成本，这个问题是一个普通的运输模型。

7.5.1 数学模型

容量限制的工厂选址问题 (capacitated plant location problem)(cPLP)被称为使用最小的总成本，包括生产、运输和工厂定址的固定成本，确定工厂位置的固定成本问题。在这种情况下，m 个源点(或设备地址)为 n 位顾客生产单一的产品，每位需求量为 $b_j(j=1,\cdots,n)$ 个单位。如果某个特殊源点 i 是开放的(或装有设备)，则它存在固定成本 $d_i\geqslant 0$ 和相关的生产能力 $a_i>0$。从源点 i 到顾客 j 也有一个正的单位运输成本 c_{ij}。问题是确定工厂的位置，使得在最小的成本下，容量不能超过，并且需求必须满足。cPLP 可以构造为如下的混合整数规划：

$$\text{cPLP:}\qquad \min\quad z(\boldsymbol{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} d_i y_i \tag{7.32}$$

$$\text{s. t.}\quad \sum_{i=1}^{m} x_{ij} = b_j,\qquad j=1,2,\cdots,n \tag{7.33}$$

$$\sum_{j=1}^{n} x_{ij} \leqslant a_i y_i,\quad i=1,2,\cdots,m \tag{7.34}$$

$$x_{ij}\geqslant 0,\qquad \forall\, i,j \tag{7.35}$$

$$y_i = 0 \text{ 或 } 1,\qquad i=1,2,\cdots,m \tag{7.36}$$

变量 x_{ij} 和 y_i 分别代表从工厂 i 到仓库 j 的运输量，工厂是开放的(或定址)($y_i=1$)或是关闭的($y_i=0$)。

目标函数(7.32)是总运输成本加总固定成本。当 $y_i=1$ 或工厂 i 开放时，d_i 仅对目标函数有贡献。约束条件(7.33)保证了每个顾客的需求得到满足。不等式(7.34)确保我们运输的产品不会来自关闭的工厂，同时，也限制了容量的超出。

当每个工厂能满足所有顾客的需求时，容量限制的工厂选址的问题(7.32)～(7.36)可以重新描述。称为无容量限制的工厂选址问题(uPLP)，可用观察法求解。为了转化问题，定义 g_{ij} 是工厂 i 满足顾客 j 的需求的份额，即：

$$g_{ij} = \frac{x_{ij}}{b_j},\quad \forall\, i,j \tag{7.37}$$

在直接使用等式(7.37)前，我们首先注意到当 $a_i\geqslant\sum_{j=1}^{n} b_j(i=1,\cdots,m)$，不等式(7.34)对于确保运输不来自关闭的工厂仅仅是必要的。这就允许我们用下面的公式代替约束条件(7.34)。

$$\sum_{j=1}^{n} g_{ij} \leqslant n y_i, \quad i = 1, \cdots, m \tag{7.38}$$

因为 $y_i = 0$ 意味着 $g_{ij} = 0 (i = 1, \cdots, m)$，没有需求会因为工厂 i 而得到满足。在表达式(7.32)~(7.34)中的 x_{ij} 可用 $g_{ij} b_j$ 代替，令 $t_{ij} = c_{ij} b_j$，则我们得到等价的工厂选址问题：

$$\text{uPLP:} \quad \min \quad z(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} g_{ij} + \sum_{i=1}^{m} d_i y_i \tag{7.39}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} g_{ij} = 1, \quad j = 1, 2, \cdots, n \tag{7.40}$$

$$\sum_{j=1}^{n} g_{ij} \leqslant n y_i, \quad i = 1, 2, \cdots, m \tag{7.41}$$

$$g_{ij} \geqslant 0, \quad \forall i, j \tag{7.42}$$

$$y_i = 0 \text{ 或 } 1, \quad i = 1, 2, \cdots, m \tag{7.43}$$

变量为 g_{ij} 和 y_i。这类问题可以通过观察法求解[171]。用 $0 \leqslant y_i \leqslant 1 (i = 1, 2, \cdots, m)$ 代替 $y_i = 0$或 1。因为工厂是无容量限制的，所以所有顾客 j 的需求由最近的开放的工厂满足。分配变量 g_{ij} 自然会被赋予整数值。在 2.3 节中介绍的装箱模型，属于 NP 难题，是当所有 i 和 j 的系数 $t_{ij} = 0$ 时容量限制的工厂选址的问题的特殊情况。

这类问题的一个重要属性是最优解必定出现在可行域的一个极点上，也就是说，它必定是与凸多边形的基本可行解相关。这意味着寻找一个全局最优解可以限制于仅需考虑一个凸多边形的极点。这个结果与线性规划的结果相似。尽管如此，因为目标函数是凹的，所以相关的算法更为复杂。

对于工厂选择问题，有各种各样的算法和丰富的文献。在建立目标和约束函数的细节上是多样的，但基本思想保持不变。因为在固定费用问题中，对于工厂问题有近似和精确算法[608]。

Efroymson 和 Ray[171]研究了针对无容量限制的工厂选址问题的分支定界算法。Davis 和 Ray[48]开发了针对容量限制的工厂选址问题的分支定界算法。分支定界算法的效率很大程度上依赖线性规划问题如何很快地被求解。Sá 提出了 cPLP 近似解的启发式技术。尽管启发式方法一般能产生优秀的解，但他们依赖大量的数据，且不能总是运行有效的。

7.5.2 针对工厂问题的基于生成树的遗传算法

在进化过程中，除了不同的评价函数外，针对容量限制的工厂选址问题，基于生成树的遗传算法与针对固定费用运输问题的基于生成树的遗传算法相同。

近来，Gen，Choi 和 Tsujimura[715]提出了基于两种表示方案的遗传算法，用于求解具有单个源约束的容量限制的工厂选址问题。

7.5.3 数例

运用基于生成树的遗传算法，试验在 HP 9000 Model 715/100 工作站 UNIX 操作系统下进行。使用解的质量和 CPU 时间来评价程序的性能。用 $m\times n$ 表示大小的三个问题的规模分别为 5×10,10×15 和 10×20。每个测试问题是随机产生的。供应量、需求量和固定成本的范围见表 7.11。表 7.12 给出了 5×10 的测试问题的随机产生的取样数据。

表 7.11 cPLP 中需求和固定成本的容量与范围

问题规模	工厂容量	需 求	固定成本
5×10	5000	[100,1000]	[200,500]
10×15	10 000	[100,2000]	[500,1000]
10×20	10 000	[200,2000]	[500,1000]

表 7.12 5×10 测试问题的取样数据

工厂 i	固定成本 d_i	交货成本 c_{ij}										供应量 a_i
1	304	10	9	8	7	4	6	4	7	8	4	1113
2	381	9	3	6	9	6	9	8	4	4	5	1247
3	459	3	5	8	7	3	10	10	8	5	4	902
4	292	6	6	3	10	8	7	4	4	6	3	912
5	241	10	10	9	4	8	5	3	10	4	6	826
需求量	b_j	729	630	321	293	251	573	207	732	481	783	5000

使用给定的遗传参数：变异概率 $p_m=0.4$、杂交概率 $p_c=0.2$，st-GA 计算运行了10 次。对于 5×10 和 10×15 两个问题，最大世代数为 2000；对于 10×20 的问题，最大世代数为 1000。对于 5×10,10×15 和 10×20 的测试问题，种群大小分别为 100,150 和 150。

将表 7.13 中所示的计算结果与基于矩阵的遗传算法(m-GA)的结果进行了比较。任何一种算法得到的最优解被作为计算解的质量的基础。最佳结果是 100%，其他解则低于 100%。解的质量定义如下：

$$\text{解的质量}=\frac{\text{最优解}}{\text{得到的解}}\times 100$$

从表 7.13 我们可以看到较之 m-GA，st-GA 找到最佳解的次数更多。比较解的质量，st-GA 也比 m-GA 更有可能进化为最优解和近似最优解。表 7.14 显示：对于小规模问题，st-GA 比 m-GA 花费的计算时间更少。对于大规模问题，尤其是对 10×20 的测试问题，st-GA 需要的 CPU 时间与 m-GA 的计算时间没有太大的差别。这是因为在初始化过程中，st-GA 需要更多的时间产生可行染色体。

表 7.13 m-GA 和 st-GA 之间解质量的比较

问题大小	m-GA				st-GA			
	最坏	平均	最优	Freq.	最坏	平均	最优	Freq.
5×10	92.43	97.02	100.00	2	100.00	100.00	100.00	10
10×15	90.48	93.10	98.97	0	96.50	98.92	100.00	3
10×20	89.16	93.62	97.84	0	92.64	96.47	100.00	1

标记"Freq."为找到最佳解的次数。

表 7.14 m-GA 和 st-GA 之间 CPU 时间的比较

问题大小	染色体的存储单元		平均 CPU 时间/min	
	m-GA	st-GA	m GA	st-GA
5×10	50	13	12.075	3.456
10×15	150	23	78.240	9.564
10×20	200	28	103.963	71.625

图 7.12 显示了取样数据为 5×10 的问题在进化过程中的平均解的质量。很显然，基于生成树的编码比基于矩阵的编码具有更好的性能，在进化过程中进化为最优解。

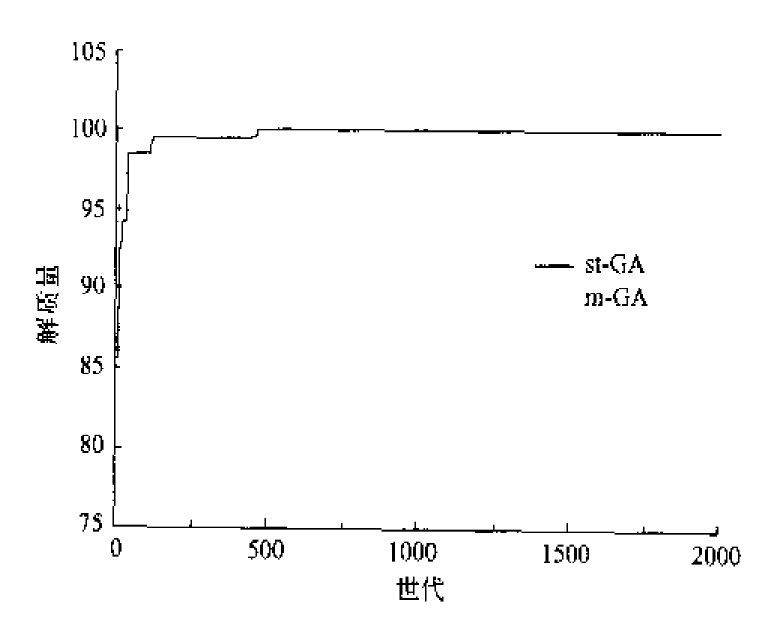


图 7.12 5×10 cPLP 的进化过程

7.6 带模糊系数的双目标运输问题

现实世界的情况往往不像我们正在讨论的那样是确定的。精确的数学模型不足以处理所有的实际问题。为了处理不精确性和不确定性，通常采用概率论的概念和技术。这

些具有不确定情况的普遍问题是难以给模型参数的确定合适的值。模糊集理论已应用于线性和非线性规划、整数规划、多目标决策等。它有助于改善过分简单的模型，为现实世界的复杂系统提供更为健壮和柔韧的模型。在考虑实际模型时，使用模糊数来表示不精确的情况是必要的。在决策中处理这样的不确定的方法之一是模糊数学规划。Kaufmann 和 Gupta[349]首先研究考虑模糊系数(fuzzy coefficients)的模糊运输问题。Gen，Li 和 Ida[412,414,714]提出了改善的遗传算法用于求解带有模糊数 (fuzzy numbers)的多目标的单一运输问题。在这节中我们使用基于生成树的遗传算法以求解带有模糊系数的双目标的运输问题 (bicriteria transportation problem，BTP)。

7.6.1 问题的表述

经常地，在运输系统中交通复杂的影响导致了一些或所有目标系数的不确定性。例如运输成本、交货时间等，不可能被明确地知道。

考虑以下两个目标：最小化总成本和最小化交货时间。$\tilde{c}_{ij}^1$是代表从工厂 i 到仓库 j 单位运输成本的模糊数据，$\tilde{c}_{ij}^2$是代表从工厂 i 到仓库 j 运输一个单位产品的交货时间的模糊数据，a_i 是在工厂 i 的产品可用量，b_j 是仓库 j 的需求量。m 家工厂 n 个仓库的这类问题可以用如下公式表示：

$$\min \quad \tilde{z}_1 = \sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{c}_{ij}^1 x_{ij} \tag{7.44}$$

$$\min \quad \tilde{z}_2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{c}_{ij}^2 x_{ij} \tag{7.45}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} \leqslant a_i, \quad i = 1,2,\cdots,m \tag{7.46}$$

$$\sum_{i=1}^{m} x_{ij} \geqslant b_j, \quad j = 1,2,\cdots,n \tag{7.47}$$

$$x_{ij} \geqslant 0, \qquad \forall i,j \tag{7.48}$$

其中，x_{ij}是从工厂 i 到仓库 j 的未知运输量。

7.6.2 排序模糊数

Liou 和 Wang 的方法用于对模糊数进行排序[424]。使用积分值代替相对值，这种方法适用于三角形模糊数 (triangular fuzzy number)或梯形模糊数。左边的积分值反映了决策者悲观的观点，右边的积分值反映了决策者乐观的观点。使用乐观指数，左、右两个积分值的凸组合被称为总积分值。

$I_L(\widetilde{A})$和 $I_R(\widetilde{A})$分别代表三角模糊数 $\widetilde{A}=(a_1,a_2,a_3)$的左积分值和右积分值。决策者的乐观指数为 α 的模糊 $\widetilde{A}$ 的总积分值为

$$I_T^\alpha(\widetilde{A}) = \alpha I_R(\widetilde{A}) + (1-\alpha)I_L(\widetilde{A}) = \frac{1}{2}[\alpha a_1 + a_2 + (1-\alpha)a_3] \tag{7.49}$$

其中，$\alpha\in[0,1]$。α 值高表明较高的乐观度[424]。

特别地，当 $\alpha=0$ 时，总积分值 $I_T^0(\widetilde{A})$ 代表悲观决策者的观点，即 $I_T^0(\widetilde{A})$ 等于 $\widetilde{A}$ 的左积分值[即 $I_L(\widetilde{A})$]。对于乐观的决策者(即 $\alpha=1$)，总积分值 $I_T^1(\widetilde{A})$ 等于 $I_R(\widetilde{A})$。对于折中的决策者($\alpha=0.5$)，总积分值 $I_T^{0.5}(\widetilde{A})=\dfrac{1}{2}[I_R(\widetilde{A})+I_L(\widetilde{A})]$，在这种情况下积分值如同通常的描述[349]。

这个模糊数总积分值被用作函数排序。对于任何模糊数 $\widetilde{A}_i$ 和 $\widetilde{A}_j$，我们有下面的准则对模糊数进行排序：

1. 如果 $I_T^\alpha(\widetilde{A}_i)<I_T^\alpha(\widetilde{A}_j)$，则 $\widetilde{A}_i<\widetilde{A}_j$。
2. 如果 $I_T^\alpha(\widetilde{A}_i)=I_T^\alpha(\widetilde{A}_j)$，则 $\widetilde{A}_i=\widetilde{A}_j$。
3. 如果 $I_T^\alpha(\widetilde{A}_i)>I_T^\alpha(\widetilde{A}_j)$，则 $\widetilde{A}_i>\widetilde{A}_j$。

该方法的详细内容可参见文献[424]。

在模糊环境中，我们定义如下的 Pareto 最优解。

定义 7.1 解 $\bar{x}=[\bar{x}_{ij}]$ 是模糊 BTP 的 Pareto 最优解，当且仅当不存在任何其他的 $x\in F$ 使得：

$$R(\tilde{z}_q(x))\geqslant R(\tilde{z}_q(\bar{x}))\quad \forall q$$
$$R(\tilde{z}_p(x))\neq R(\tilde{z}_p(\bar{x}))\quad \exists p$$

其中，$R(\cdot)$ 称作排序函数，$\tilde{z}_q$ 被最小化的第 q 个目标函数值的模糊数。

模糊环境中的 Pareto 最优解是基于模糊目标的排序值确定的。人们提出了几种模糊数的排序方法[349,424]。在这个问题中，模糊数的积分值被用作排序函数。

7.6.3 遗传算法的实现

基于生成树的遗传算法被用于求解这个问题。基本的操作与第 4 章中给出的相似。绝大多数的工作被用于处理模糊。在多目标最优化中，我们对寻找 Pareto 最优解感兴趣。当目标中的系数用模糊数代替时，目标值就变成了模糊值。因为，模糊值代表所有可能的实数值，所以并不容易比较解以确定哪一个是 Pareto 最优解。模糊排序技术可以帮助我们比较模糊数。在这种方法中，基于模糊目标函数的排序值确定 Pareto 最优解，遗传算法被用于寻找 Pareto 最优解。

遗传子表示 基于生成树的 Prüfer 数编码被用于表示候选解。编码和解码的过程在 7.2 节中已描述。同时也采用了在基于生成树的遗传算法中设计的解的可行性准则。

遗传运算：

杂交 简单地，使用单分割点杂交。

变异 使用反转变异和位移变异。两类变异操作被随机选择用于给定染色体的变异运算。

评价与选择 在这种方法中，评价过程由两步组成：

1. 将染色体转化为树。

2. 计算带模糊系数的每个目标函数。

在目标函数中，系数用模糊数表示。目标函数的计算由下面的表达式代替。

$$\tilde{z}_1(T) \leftarrow \tilde{z}_1(T) + \tilde{c}_{ij}^1 x_{ij}$$

$$\tilde{z}_2(T) \leftarrow \tilde{z}_2(T) + \tilde{c}_{ij}^2 x_{ij}$$

Pareto 最优解作为多目标规划问题的解被描述。因此，获取 Pareto 最优解的模块被嵌入到遗传算法中，这包含以下两步：

1. 使用目标函数评价染色体。

2. 基于评价值选择 Pareto 最优解。

设 E 是到当前世代 t 为止产生的 Pareto 解集，下面给出求解 Pareto 解的过程：

过程 Pareto 解的求解过程

第 1 步：设置迭代标志 $k=1, E=\{\varnothing\}$。

第 2 步：如果 $k>i_size$，则停止；否则，执行第 3 步。

第 3 步：评价染色体 T_k，得到解向量 $\tilde{z}_k=[\tilde{z}_1(T_k)\tilde{z}_2(T_k)\cdots\tilde{z}_Q(T_k)]$。

第 4 步：将它与 E 中所有 Pareto 解进行比较。

(4.1) 使用模糊数的排序方法，基于解向量的排序值，如果有一个 Pareto 解优于它，执行第 5 步。

(4.2) 使用模糊数的排序方法，基于解向量的排序值，如果它优于部分 Pareto 解，则将其增加入 E，删除 E 中被其优超的解。

(4.3) 如果它是一个新 Pareto 解，不优于 E 中任何解，则简单地将其添加入 E。

第 5 步：设置 $k=k+1$，返回第 2 步。

对于进化过程中染色体的适应值，采用下面的过程来创建适应值函数。

适应值函数 适应值函数按下面的方式推导：

1. 选择包含对应每个目标函数值的最小值 $I_T^\alpha(\tilde{z}_q^{\min})$[或最大值 $I_T^\alpha(\tilde{z}_q^{\max})$]的解点，与先前世代保存下来的解进行比较，再次选择最佳点并保存。

$$I_T^\alpha(\tilde{z}_q^{\min(t)}) = \min_k\{I_T^\alpha(\tilde{z}_q^{\min(t-1)}), I_T^\alpha(\tilde{z}_q^{(t)}(T_k)) \mid k=1,2,\cdots,i_size\}, \quad q=1,2$$

$$I_T^\alpha(\tilde{z}_q^{\max(t)}) = \max_k\{I_T^\alpha(\tilde{z}_q^{\max(t-1)}), I_T^\alpha(\tilde{z}_q^{(t)}(T_k)) \mid k=1,2,\cdots,i_size\}, \quad q=1,2$$

2. 建立适应值函数，得到每个目标函数的权重：

$$\delta_q = I_T^\alpha(\tilde{z}_q^{\max(t)}) - I_T^\alpha(\tilde{z}_q^{\min(t)}), \quad q=1,2$$

$$\beta_q = \frac{\delta_q}{\sum_{q=1}^{2}\delta_q}, \quad q=1,2$$

$$\mathrm{eval}(T_k)=\sum_{q=1}^{2}\beta_q I_T^{\alpha}(\tilde{z}_q(T_k)),\quad \forall k$$

整个算法过程　令 $P(t)$是当前世代 t 时的染色体种群，$C(t)$是当前世代 t 产生的染色体，$E(t)$是到当前世代 t 为止产生的 Pareto 解的集合。整个算法过程归结如下：

```
procedure: st-GA/b-fTP
begin
    t ← 0;
    初始化 P(t);
    应用适应值函数评价 P(t);
    确定 Pareto 解集 E(t);
    while 终止条件不满足 do
    begin
        重组 P(t)以产生 C(t);
        用适应值函数评价 C(t);
        更新 Pareto 解集 E(t);
        从 P(t)和 C(t)中选择 P(t+1);
        t ← t+1;
    end
end
```

7.6.4　数例

对下面的问题进行了计算模拟，问题的系数由表 7.15 给出的三角形模糊数(TFNs)描述。

表 7.15　第一个例子中具有 3 家工厂 4 个仓库的 TFN 系数

目标 k	$1:\tilde{c}_{i1}^k$	$2:\tilde{c}_{i2}^k$	$3:\tilde{c}_{i3}^k$	$4:\tilde{c}_{i4}^k$	a_i
1	(1,2,3)	(1,2,3)	(5,7,9)	(6,7,9)	8
	(1,1,2)	(6,9,12)	(2,3,5)	(3,4,5)	19
	(6,8,10)	(7,9,10)	(2,4,6)	(5,6,8)	17
2	(3,4,5)	(2,4,6)	(1,3,5)	(2,4,6)	8
	(3,5,7)	(6,8,9)	(7,9,9)	(8,9,12)	9
	(4,6,8)	(1,2,3)	(4,5,6)	(1,3,3)	17
b_j	11	3	14	16	44

使用最佳的参数设置(杂交概率 $p_c=0.2$，变异概率 $p_m=0.5$)，种群大小为 30，最大迭代世代数为 1000，以及三个不同的乐观度：$\alpha=0$，0.5 和 1，设计的算法运行 30 次，其计算结果见图 7.13。计算结果可在 Pareto 前沿面上找到。基于每个目标函数值的积分值

得到 Pareto 解，在悲观情况下得到的 8 个 Pareto 解如下：

(1) 解矩阵：$\begin{bmatrix} 5 & 3 & 0 & 0 \\ 6 & 0 & 0 & 13 \\ 0 & 0 & 14 & 3 \end{bmatrix}$

它相应的目标函数值是 $\tilde{z}_1=(96,148,209)$，$\tilde{z}_2=(202,258,334)$。目标值的积分值 $(I_T^0(\tilde{z}_1),I_T^0(\tilde{z}_2))=(74.00,129.00)$，$P(T)=[6\ 1\ 4\ 2\ 2]$。

(2) 解矩阵：$\begin{bmatrix} 0 & 3 & 0 & 5 \\ 11 & 0 & 0 & 8 \\ 0 & 0 & 14 & 3 \end{bmatrix}$

它相应的目标函数值是 $\tilde{z}_1=(111,158,224)$，$\tilde{z}_2=(172,238,314)$。目标值的积分值 $(I_T^0(\tilde{z}_1),I_T^0(\tilde{z}_2))=(79.00,119.00)$，$P(T)=[6\ 2\ 1\ 6\ 2]$。

(3) 解矩阵：$\begin{bmatrix} 5 & 3 & 0 & 0 \\ 6 & 0 & 13 & 0 \\ 0 & 0 & 1 & 16 \end{bmatrix}$

它相应的目标函数值是 $\tilde{z}_1=(122,161,235)$，$\tilde{z}_2=(150,232,256)$。目标值的积分值 $(I_T^0(\tilde{z}_1),I_T^0(\tilde{z}_2))=(80.5,116)$，$P(T)=[1\ 4\ 2\ 6\ 3]$。

(4) 解矩阵：$\begin{bmatrix} 0 & 3 & 0 & 5 \\ 11 & 0 & 8 & 0 \\ 0 & 0 & 6 & 11 \end{bmatrix}$

它相应的目标函数值是 $\tilde{z}_1=(127,166,240)$，$\tilde{z}_2=(140,222,266)$。目标值的积分值 $(I_T^0(\tilde{z}_1),I_T^0(\tilde{z}_2))=(83,111)$，$P(T)=[5\ 2\ 2\ 6\ 3]$。

(5) 解矩阵：$\begin{bmatrix} 0 & 3 & 5 & 0 \\ 11 & 0 & 8 & 0 \\ 0 & 0 & 1 & 16 \end{bmatrix}$

它相应的目标函数值是 $\tilde{z}_1=(137,176,250)$，$\tilde{z}_2=(120,207,246)$。目标值的积分值 $(I_T^0(\tilde{z}_1),I_T^0(\tilde{z}_2))=(88,103.5)$，$P(T)=[7\ 2\ 1\ 6\ 2]$。

(6) 解矩阵：$\begin{bmatrix} 0 & 2 & 6 & 0 \\ 11 & 0 & 8 & 0 \\ 0 & 1 & 0 & 16 \end{bmatrix}$

它相应的目标函数值是 $\tilde{z}_1=(146,186,260)$，$\tilde{z}_2=(116,203,242)$。目标值的积分值 $(I_T^0(\tilde{z}_1),I_T^0(\tilde{z}_2))=(93,101.5)$，$P(T)=[2\ 6\ 1\ 5\ 3]$。

(7) 解矩阵：$\begin{bmatrix} 0 & 0 & 6 & 2 \\ 11 & 0 & 8 & 0 \\ 0 & 3 & 0 & 14 \end{bmatrix}$

它相应的目标函数值是 $\tilde{z}_1=(160,202,276)$，$\tilde{z}_2=(116,201,242)$。目标值的积分值 $(I_T^0(\tilde{z}_1),I_T^0(\tilde{z}_2))=(101,100.5)$，$P(T)=[5\ 2\ 2\ 6\ 1]$。

(8) 解矩阵：

$$\begin{bmatrix} 0 & 0 & 8 & 0 \\ 11 & 0 & 6 & 2 \\ 0 & 3 & 0 & 14 \end{bmatrix}$$

它相应的目标函数值是 $\tilde{z}_1=(160,204,276)$，$\tilde{z}_2=(116,199,246)$。目标值的积分值 $(I_T^0(\tilde{z}_1),I_T^0(\tilde{z}_2))=(102,99.5)$，$P(T)=[4\ 3\ 4\ 1\ 1]$。

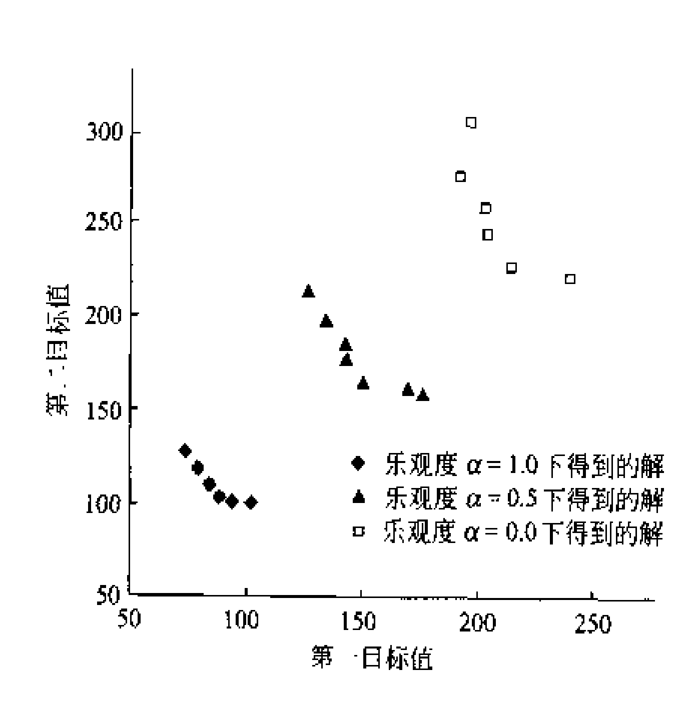


图 7.13　第一个例子中三个乐观度下得到的解

第二个例子有 6 家工厂 9 个仓库。表 7.16 给出由三角模糊数描绘的双目标的系数。经过测试，最佳的遗传参数设置为：杂交概率 $p_c=0.2$，变异概率 $p_m=0.5$，运行次数为 30 次。图 7.14 比较了乐观、折中、悲观三个程度下的结果。计算结果表明当决策者以乐观的态度评价运输项目($\alpha=1$)，目标值是该问题最小的。因此决策者可以在不精确的情况下得到其所期望的目标值的范围。

表 7.16　第二个例子中具有 6 家工厂 9 个仓库的 TFN 系数

k	$1:\tilde{c}_{i1}^k$	$2:\tilde{c}_{i2}^k$	$3:\tilde{c}_{i3}^k$	$4:\tilde{c}_{i4}^k$	$5:\tilde{c}_{i5}^k$	$6:\tilde{c}_{i6}^k$	$7:\tilde{c}_{i7}^k$	$8:\tilde{c}_{i8}^k$	$9:\tilde{c}_{i9}^k$	a_i
1	(1,1,3)	(1,2,3)	(5,7,9)	(6,7,9)	(2,4,5)	(6,7,8)	(4,5,6)	(1,3,5)	(7,8,9)	8
	(1,1,2)	(6,9,12)	(2,3,5)	(3,4,5)	(7,9,10)	(6,7,8)	(2,4,6)	(3,5,8)	(5,6,7)	19
	(6,8,10)	(7,9,10)	(2,4,6)	(5,6,8)	(5,6,8)	(1,4,5)	(6,9,11)	(3,5,7)	(8,9,12)	17
	(5,6,7)	(1,4,5)	(8,9,11)	(3,5,7)	(3,5,7)	(6,8,10)	(7,9,9)	(8,9,12)	(9,11,13)	23
	(2,4,5)	(6,7,8)	(4,5,6)	(1,3,5)	(10,13,15)	(8,9,10)	(2,5,7)	(4,7,9)	(7,8,9)	10
	(7,9,11)	(3,4,6)	(2,5,7)	(4,7,9)	(7,8,9)	(5,7,9)	(9,12,15)	(3,4,6)	(5,8,10)	20
2	(3,4,5)	(2,4,6)	(1,3,5)	(2,4,6)	(10,13,15)	(8,9,10)	(2,5,7)	(4,7,9)	(7,8,9)	8
	(3,5,7)	(6,8,9)	(7,9,9)	(8,9,12)	(1,4,5)	(8,9,11)	(3,5,7)	(3,5,7)	(6,8,10)	19
	(4,6,8)	(1,2,3)	(4,5,6)	(1,3,3)	(6,9,12)	(2,3,5)	(8,9,10)	(2,5,7)	(4,7,9)	17
	(6,8,10)	(7,9,10)	(2,4,6)	(5,6,8)	(9,12,15)	(3,4,6)	(5,8,10)	(8,9,11)	(3,5,7)	23
	(5,6,7)	(1,4,5)	(8,9,11)	(3,5,7)	(5,7,9)	(9,12,15)	(3,4,6)	(5,8,10)	(3,4,6)	10
	(2,4,6)	(6,7,8)	(4,5,6)	(1,3,5)	(4,7,9)	(7,8,9)	(5,7,9)	(9,12,15)	(3,4,6)	20
b_j	11	3	14	16	8	6	10	20	9	

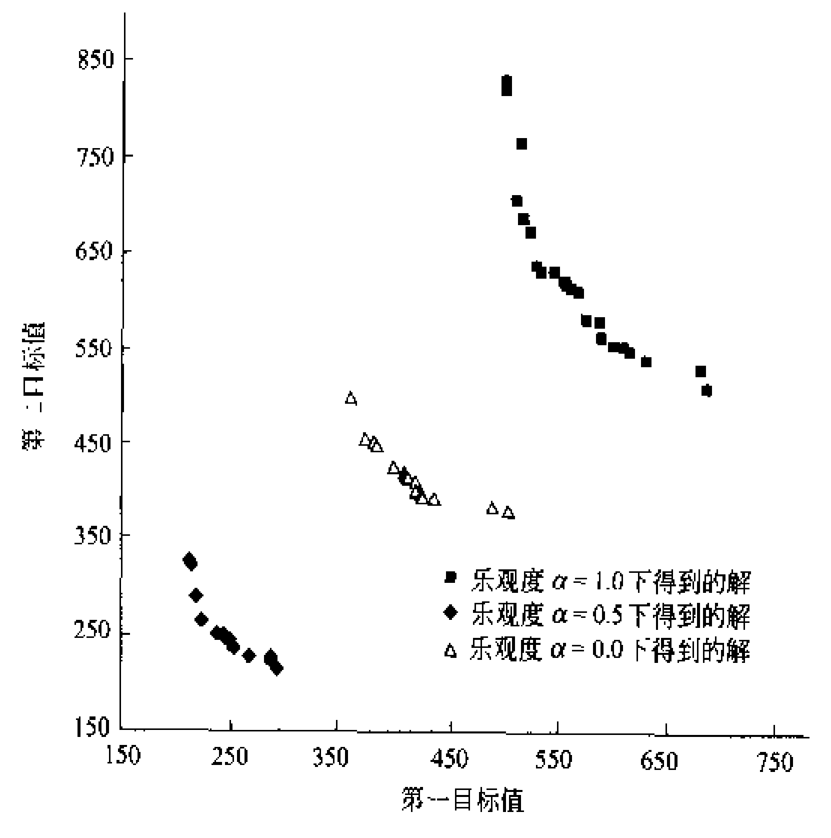


图 7.14　第二个例子中三个乐观度下得到的解

第8章 网络设计与路径

8.1 引言

自从20世纪80年代以来，计算机网络(computer networks)得到了爆炸性的发展。从自动取款机，到机票预订系统，电子邮件服务器，电子公告牌，因特网，计算机网络已渗入到我们日常生活的方方面面。计算机网络的爆炸性发展有许多原因，我们正处在信息时代，计算机网络已成为信息传播不可缺少的一部分。计算机网络的设计与路由(routing)是构建和扩展计算机网络的重要内容之一。在过去的20年中，人们提出并测试了许多思想和方法。近年来，将遗传算法应用于与计算机网络相关的问题，正日益激起人们的兴趣[85,89,175,210,212,380,472,474,558,642,643]。在这一章中，我们将解释如何运用遗传算法解决网络系统中的一些问题，包括最短路径问题(shortest path problem)、集中式网络(centralized network)的有适应能力的路由设计，以及网络上设施的定位问题[228,241,474]。

8.2 最短路径问题

最短路径问题是网络设计中遇到的最常见问题之一：寻找指定两节点间总长度最短或费用最低的路径。在许多实际应用中，包括运输、路径、通信，这是一个基本问题[528]。然而，在许多应用中，网络中的每条边都与一些评价准则有关，例如：费用和时间在运输网络中都很重要，正如在公路建设中的经济和生态因素一样。因此，近期的研究兴趣是求解双目标最短路径问题(bicriteria shortest-path problem)，即寻找对于两个目标都是有效的路径。通常，不存在一条路径，能同时满足两个目标的最短路径。相反，只存在一组Pareto最优路径或有效路径。这类问题的一些应用在文献[293]和[656]中已详细记载。

有一些算法可以用于寻找两目标条件下的有效路径[125,282,293,442,656]。一般而言，这些算法与产生Pareto最优解的方法有着共同的特点，需要判断整个有效解集或者这种解集足够大的子集。Smith和Shier给出了这些方法的经验调查结果[577]。

两目标的最短路径问题以NP-难题著称[208]。有效的路径集可能很庞大，可能与问题规模成指数关系。因此，在最坏情况下，求解问题所需要的计算量可能会随问题大小呈指数增长[282]。当求解大规模问题时，问题的易处理性是十分重要的，这对于决策者而言，有效解集的大小是重要的。为了选择最佳答案而不得不评价一个大型的有效解集，是决

策者必须承担的一个认知上相当大的负担。因此，在这样的情况下，寻求 Pareto 最优解集几乎没有意义。Murthy 和 Olson 提出了一套交互式的程序，以引导决策者快速得到他们的首选解[479]。

Cheng 和 Gen 提出了一套基于遗传算法的折中方法，以求解两目标的最短路径问题[109]。与有效解生成法比较，折中的方法通过一些对距离的测量将最接近理想解的解视为寻找的解。然而，应用遗传算法求解这类问题的难点在于如何将图中的路径编码成染色体。一种基于优先权的编码方法可以潜在地表示图中所有可能路径。此外，基于优先权的编码方法的最显著特征之一是，它具有表示非平面无向图中路径的能力。

8.2.1 问题描述

无向图 $G=(V,E)$ 由一组点 $V=\{1,2,\cdots,n\}$ 和一组连接 V 中节点的边 $E\in V\times V$ 组成。对应每条边有两个非负数 c_{ij}^1 和 c_{ij}^2 分别代表从节点 i 到 j 的费用和距离，或其他效益指标。从节点 i 到 j 的路径是 E 中的一个边序列 $(i,l),(l,m),\cdots,(k,j)$。路径中没有一个节点出现的次数多于 2。路径也可以用一个节点序列 $(i,l,m,\cdots,k,j)$ 等价表示。如图 8.1 所示：(1,4)，(4,3)，(3,5)，(5,6) 是从节点 1 到 6 的路径，用节点序列表示为 (1,4,3,5,6)。

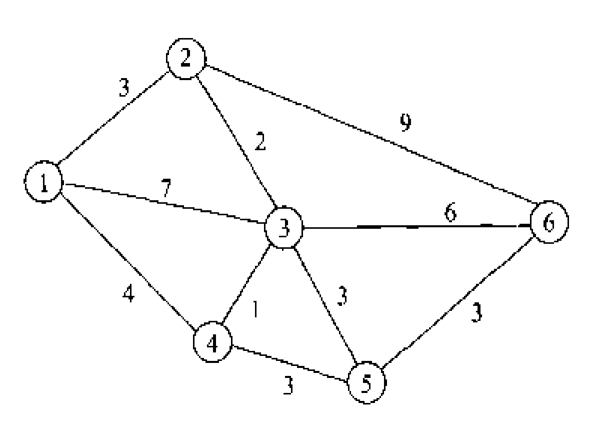


图 8.1 6 个节点和 10 条边的简单无向图

令 1 表示路径的初始节点，n 表示路径的终节点，x_{ij} 是如下定义的指示变量：

$$x_{ij}=\begin{cases}1, & \text{如果边}(i,j)\text{ 在路径中}\\ 0, & \text{其他}\end{cases}$$

两目标最短路径问题可以表示如下：

$$\min \quad z^1(\boldsymbol{x})=\sum_i\sum_j c_{ij}^1 x_{ij} \tag{8.1}$$

$$\min \quad z^2(\boldsymbol{x})=\sum_i\sum_j c_{ij}^2 x_{ij} \tag{8.2}$$

$$\text{s.t.} \quad \sum_j x_{ij}\leqslant 2, \qquad \forall i\in V \tag{8.3}$$

$$\sum_{j\neq k} x_{ij}\geqslant x_{ik}, \qquad \forall (i,k)\in E, \quad \forall i\in V\backslash\{1,n\} \tag{8.4}$$

$$\sum_j x_{1j}=\sum_j x_{jn}=1, \qquad \forall i,j\in V \tag{8.5}$$

$$x_{ij} = x_{ji}, \qquad \forall (i,j) \in E \tag{8.6}$$

$$0 \leqslant x_{ij} \leqslant 1, \qquad \forall (i,j) \in E \tag{8.7}$$

其中约束条件(8.3)和(8.4)共同确保除节点1和n外的任何一个节点，不存在或只有两条相邻的边。约束条件(8.5)确保节点1和n是路径的端点。

8.2.2 遗传算法的方法

基于优先权的编码(priority-based encoding)：开发遗传算法解决最短路径问题，为图中的路径编码是至关重要的。寻找这一问题的遗传子表示方法并不比旅行商问题简单。产生特殊困难的原因是：

1. 路径包含节点数可以是变化的，对于n个节点图的路径的最大数为$n-1$。
2. 边的随机序列通常与路径不一致。

为了克服这些困难，Cheng和Gen采用了一种间接的方法：对路径的导向性信息进行编码，构建染色体中的路径而不是路径本身。

众所周知，染色体中的基因由两个要素表征：基因点，在染色体结构中基因所处的位置；基因值，基因携带的值。基因的位置可以代表节点，基因值代表候选点中组成路径的节点的优先权。这种编码方法称作基于优先权的编码[111]。对应于给定的染色体的路径由一系列始于节点1，终于节点n的相继节点构成。在每一步中，通常有几个节点可供考虑，但只有优先权最高的节点加入路径中。

考虑图8.2中的无向图。假设要寻找一条节点1到10的路径。在这个实例中，编码如图8.3所示。开始，我们试图寻找与节点1相邻的节点。节点2和3在此都是适合的，根据节点间的连接关系这是很容易确定的。它们的优先权分别为3和4，节点4具有更高的优先权，因此，被纳入路径。然后我们得到了下一位置的可行节点集，从中选择优先权最高的一条。重复这些步骤，直至得到一条完整的路径(1,3,6,7,8,10)。

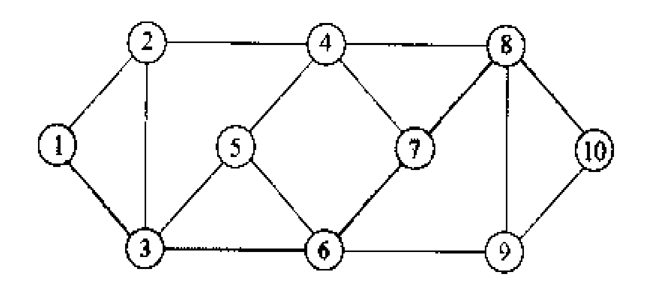


图8.2 10个节点和16条边的简单无向图

位置：节点ID	1	2	3	4	5	6	7	8	9	10
值：优先权	7	3	4	6	2	5	8	10	1	9

图8.3 基于优先权编码的例子

对于有n个节点的问题，令Ω是包含1到n的整数集合，即$\Omega=\{1,2,\cdots,n\}$；p_i表示节点i的优先权，它是一个Ω中的随机整数。所有节点的优先权满足下列条件：

$$p_i \neq p_j, \quad p_i, p_j \in \Omega, \quad i \neq j, \quad i,j = 1,2,\cdots,n \tag{8.8}$$

因而，基于优先权的编码形式上可以定义如下：

$$[p_1, p_2, \cdots, p_n]$$

实质上，编码与路径之间的映射是多对一的，这意味着不同的染色体可能对应着同一条路径。但是很容易证明出现多对一的概率是很低的。因此，大多数情况下，不存在与编码相关的无关重要的遗传操作。也很容易证明：任何顺序的编码对应着一条路径。因此，绝大多数已有的遗传操作可以轻易地在这种编码上运用。同样，任何一条路径拥有相应的编码。因此，遗传搜索能到达解空间中的任意一点。

路径生长过程：路径生长(path growth)过程被用于从任意的一条染色体产生路径。这一过程的基本思想是，通过不断地添加符合条件的边到路径中，产生从初始节点 1 到 n 的路径。在每一步中，通常有几条边可供考虑。添加到局部路径中的边始终是与当前节点相连接的具有最高优先权节点的边，这就从端点延长了局部路径。当然，最棘手的就是找到一个合格的边集。下面的定义和定理有助于确定这样一个合格的边集，以及路径生长过程是如何工作的。

令 $G=(V,E)$ 是定义在 E 上的具有实数赋权的连通无向图。P_t^k 是在生长中的局部路径，它包含 $k+1$ 个节点，以 t 为端点。合格边定义如下：

定义 8.1(合格边(eligible edge)) 如果一条边可以延伸路径，但不与 P_t^k 中的边构成回路，则这条边对于路径 P_t^k 是合格的。

令 $V_p \subseteq V$ 是局部路径 P_t^k 中的节点集合，$V_q = V - V_p$ 是 V_p 的补集，令 $C(V_p, V_q) = \{(i,j) | i \in V_p, j \in V_q\}$ 是图的割集[71]，$T_t = \{(t,j) | j \in V\}$ 是与端点 t 相关联的边的集合。则我们有以下性质。

性质 8.1(合格边集(eligible edge set)) 对于给定的局部路径 P_t^k，合格边集如下给出：

$$E_{qt} = T_t \cap C(V_p, V_q) \tag{8.9}$$

现在，我们考虑如何寻找 E_{qt} 集。对于连通无向图 $G=(V,E)$，邻接矩阵 $\boldsymbol{A}$ 是如下定义的 $n \times n$ 矩阵：

$$a_{ij} = \begin{cases} 1, & \text{如果}(i,j) \in E \\ 0, & \text{其他} \end{cases} \tag{8.10}$$

对于给定局部路径 P_t^k，可以定义如下的网络矩阵 $\boldsymbol{M}(k)$：

$$m_{ij} = \begin{cases} 0, & \text{如果 } i \in V, \quad \forall j \in V_p \\ 1, & \text{其他} \end{cases} \tag{8.11}$$

然后，根据给定的局部路径 P_t^k，可以定义新的邻接矩阵 $\boldsymbol{A}(k)$。

定义 8.2(动态邻接矩阵(dynamic adjacent matrix)) 对于给定局部路径 P_t^k 的动态邻接矩阵 $\boldsymbol{A}(k)$ 由下面的布尔矩阵交集给出：

$$\boldsymbol{A}(k) = \boldsymbol{A} \cap \boldsymbol{M}(k) \tag{8.12}$$

我们将看到动态邻接矩阵 $\boldsymbol{A}(k)$ 移去了所有与局部路径 P_t^k 上的节点相关联的边。也就是说，它仅包含有关局部路径 P_t^k 的邻接关系。因为，这样的邻接关系随着 P_t^k 的生长而变化，所以我们称其为动态邻接矩阵。

根据路径 P_t^k，可以如下进一步定义矩阵 $\boldsymbol{U}(k)$：

$$u_{ij} = \begin{cases} 0, & 如果\ j = t \\ 1, & 其他 \end{cases} \tag{8.13}$$

然后，我们有关于网络矩阵的如下递归方程和第 k 步的动态邻接矩阵：

$$\boldsymbol{M}(k) = \boldsymbol{M}(k-1) \cap \boldsymbol{U}(k), \boldsymbol{M}(0) = [1]_{n\times n} \tag{8.14}$$

$$\begin{aligned} \boldsymbol{A}(k) &= \boldsymbol{A}(k-1) \cap \boldsymbol{U}(k) = \boldsymbol{A}(k-1) \cap \boldsymbol{M}(k) \\ &= \boldsymbol{A} \cap \boldsymbol{M}(k), \boldsymbol{A}(0) = \boldsymbol{A} \end{aligned} \tag{8.15}$$

性质 8.2(合格边集) 令 $E(k)=\{(t,j)\mid a_{tj}(k)=1, \forall j\in V\}$，我们有

$$E(k) = E_{qt} \tag{8.16}$$

从 $\boldsymbol{A}(k)$ 很容易得到集合 $E(k)$，$\boldsymbol{A}(k)$ 采用递归的形式，容易编程实现。路径生长过程的基本思想是在每一步通过增加一条集合 $E(k)$ 中的新边延伸路径 P_t^k。运用这种方法生长的路径有可能会在某一点悬挂住，也就是说，从这一点我们将不能到达最终节点 n。这样的节点叫做悬挂点。

定义 8.3(悬挂节点(pendant node)) 对于给定的 $t\neq n$ 的局部路径 P_t^k，如果所有与端点 t 直接相连的点都属于集合 V_p，则端点 t 是悬挂点。

下面的性质给出了悬挂点的必要条件。

性质 8.3(悬挂节点) 对于给定局部路径 P_t^k，当且仅当：(1)$E(k)\neq 0$；(2)$t\neq n$ 时，端点 t 是悬挂点。

事实上，在 V_q 中存在这样一些特殊点，它们根本无法达到终点 n。如果这样的点被加入局部路径中，它必定将误导路径至悬挂节点。下面的定义和定理给了我们一种从 V_q 中区分这类节点的方法。

定义 8.4(死点(dead node)) 给定 $t\neq n$ 的局部路径 P_t^k，对于某节点 $i\in V_q$，令 $P(i)$ 表示从节点 i 到终点 n 的所有可能路径，V_{l_i} 是包含在路径 l_i 中的节点集合。如果

$$V_p \cap V_{l_i} \neq \varnothing, \quad \forall l_i \in P(i) \tag{8.17}$$

则节点 i 是死点。即，如果从该节点到节点 n 的任一可能路径至少包含 V_p 中的一个节点，则这个节点是死点。

利用可达矩阵可以判别死点[534]。令 $\boldsymbol{A}$ 是给定图 $G=(V,E)$ 的相邻矩阵。$\boldsymbol{A}$ 的 k 阶相邻矩阵可由如下的布尔乘法定义：

$$\boldsymbol{A}^k = \boldsymbol{A}^{k-1}\boldsymbol{A}, \quad \boldsymbol{A}^0 = \boldsymbol{I} \tag{8.18}$$

k 阶相邻矩阵描述了节点 i 和 j 之间的关系：两者不直接相连，但可以通过长度为 $k-1$ 的路径从节点 i 到达节点 j。如果 $a_{ij}^k=1$，则节点 i 与节点 j 是 k 阶相邻的。

令 $E_p=\{(i,j)\mid i\in V_p, j\in V\}$为与 P_i^k 上所有节点相关联的边的集合。我们可以定义图 $G(k)=(V-V_p, E-E_p)$是除去 V_p 中所有节点及与之相连的边的图 G 的子图。图 $G(k)$的可达矩阵 $\boldsymbol{R}(k)$定义如下。

定义 8.5(可达矩阵(reachability matrix)) 图 $G(k)$的可达矩阵 $\boldsymbol{R}(k)$为

$$\boldsymbol{R}(k)=[\boldsymbol{A}(k)+\boldsymbol{I}]^{n-k-2},\quad k<n-2 \tag{8.19}$$

其中,$\boldsymbol{I}$ 是单位矩阵,$\boldsymbol{A}(k)$是路径 P_i^k 的动态邻接矩阵。

可达矩阵描述了任意两点间的关系,即它们是至多 $n-k-2$ 相邻的。然后,我们有下面的性质来判别死点。

性质 8.4(死点) 对于给定局部路径 P_i^k,当且仅当 $r_{in}(k)=0$,则集合 V_q 中的节点 i 是死点。

理论上,当程序进行到每一步时,我们都可以根据性质 8.4 来检验某个节点是否是死点,并从集合 V 中除去所有的死点。对于规模大的问题,这样的检验需要消耗大量的计算成本。我们知道进化方法的优势在于:它利用惩罚机制允许不可行的染色体进入种群,与可行的染色体一起进化。在进化过程中不可行的染色体可能提供一些有用的基因,因此,不必每步都检验死点,而只是简单地通过或轻或重的惩罚,在终端死点和终点 n 之间建立一种虚拟连接。下面关于路径生长过程就是基于这些考虑的。

过程 路径生长

第 1 步:初始化。令 $k\leftarrow 0, V_p^k\leftarrow\{1\}, \boldsymbol{A}(k)\leftarrow\boldsymbol{A}, t^k\leftarrow 1$。

第 2 步:执行终止测试。如果 $t^k=n$,执行第 9 步;否则,继续。

第 3 步:确定合格边集。根据动态邻接矩阵 $\boldsymbol{A}(k)$得到边集 $E(k)$。

第 4 步:执行悬挂节点测试。如果 $E(k)=\varnothing$,令 $t^{k+1}\leftarrow n, V_p^{k+1}\leftarrow V_p^k\cup\{n\}$,给出虚拟连接 (t^k,n)的惩罚,执行第 9 步;否则,继续。

第 5 步:延伸路径。从 V_q^k 中选择优先权最高的 t^{k+1},满足$(t^k,t^{k+1})\in E(k)$。

第 6 步:执行 mesh 矩阵更新。新的 mesh 矩阵 $\boldsymbol{M}(k-1)\leftarrow\boldsymbol{M}(k)\cap\boldsymbol{U}(k+1)$。

第 7 步:执行动态邻接矩阵更新。令 $\boldsymbol{A}(k+1)\leftarrow\boldsymbol{A}(k)\cap\boldsymbol{M}(k+1)$。

第 8 步:执行迭代标记更新。$k\leftarrow k+1$,执行第 2 步。

第 9 步:返回完整的路径。

遗传操作 先前提出的编码性质是一种排列表示。在过去的 30 年中,许多重组运算应用于排列表示。这里采用的是 Syswerda 提出的基于位置的杂交(position-based crossover)运算[604],它可以看成是一种整数排列上结合修正过程的均匀杂交运算,如图 8.4 所示。事实上,它通过从左到右的扫描,从一父代上随机拿走一些基因,在空白处填入另一父

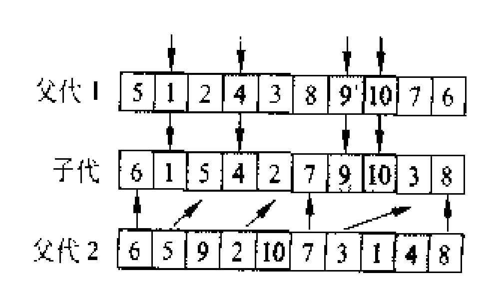


图 8.4 基于位置的杂交运算

代的基因。这里使用的是交换变异操作，随机地选择两个位置，交换它们的基因值，如图8.5所示。

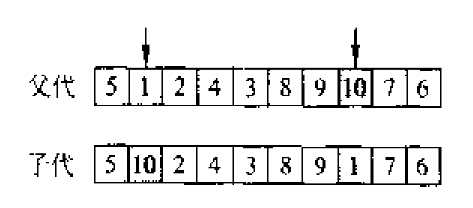


图8.5　变异运算

在遗传运算的每一次迭代中，通过对适应值的测量来评价染色体。在评价中主要有3个步骤：

1. 将染色体转化为路径。
2. 计算目标值。
3. 将目标值转化为适应值。

在选择过程中，采用了轮盘赌选择方法，这是一种与适应值成正比的选择。最优性方法与这种方法相结合，以保证最优染色体在下一世代中，避免取样的随机误差。在最优化的选择中，如果当前世代中的最佳个体不会在新的世代中再次产生，则当前世代中的一个个体被随机排除在新的种群之外，最佳个体被加入新的种群。

基于折中方法的适应值分配　正如3.2和3.8节中讨论的，折中方法是以距离函数的方式进行目标搜索的数学描述。利用下面的加权 L_p 范数，折中方法将接近最理想点的解作为最终结果：

$$r(z;p,w)=||z-z^*||_{p,w}$$

$$=\left[\sum_{j=1}^{2}w_j^p\,|z_j-z_j^*|^p\right]^{1/p}$$

其中 $z^*=(z_1^*,z_2^*)$ 是问题的理想点。参数 p 用来反映决策者的偏好，当令 $p=1$ 时，重点是所有目标的后悔值之和；当令 $p=\infty$ 时，重点是单个目标的后悔值。

对于两目标的最短路径问题，利用已有方法求解两个单目标的问题，很容易得到其理想点。对于许多复杂的问题，要找到一个理想点也是很困难的。为了解决这一困难，有人提出代理理想点的概念来替代理想点。代理理想点是当前世代的理想点，而不是给定问题的理想点。也就是说，它是在局部解空间中，而不是在整个解空间中得到的。在每一世代中，代理理想点很容易得到。沿着进化的过程，代理理想点将逐步接近真实理想点。令 P 表示当前种群的集合。代理理想点 $(z_{\min}^1, z_{\min}^2)$ 可以计算如下：

$$z_{\min}^1=\min\{z^1(x)\mid x\in P\} \tag{8.20}$$

$$z_{\min}^2=\min\{z^2(x)\mid x\in P\} \tag{8.21}$$

由于后悔值越小，个体就越优，所以必须将后悔值转化为适应值，以确保较适合的个体有更大的适应值。设 $r(x)$ 表示个体 x 的后悔值，$r_{\max}$ 是当前世代中最大的后悔值，$r_{\min}$ 是当前世代中最小的后悔值。转化公式如下：

$$\text{eval}(x)=\frac{r_{\max}-r(x)+\gamma}{r_{\max}-r_{\min}+\gamma} \tag{8.22}$$

其中 r 为受限于开区间(0,1)中的一个正实数。它的作用有二：(1)避免等式(8.22)出现分母为零；(2)可以调整选择方式从与适应值成正比的选择到纯粹的随机选择。

整个过程总结如下。

遗传算法的整个过程

第 0 步：设置遗传参数，读取给定实例的数据。

第 1 步：随机产生初始种群。

第 2 步：将染色体解码成路径。

第 3 步：计算每个解码个体的目标值。

第 4 步：计算当前世代的代理理想点。

第 5 步：计算每个个体的后悔值。

第 6 步：将后悔值转化为适应值。

第 7 步：使用轮盘赌选择的方法产生下一世代。

第 8 步：如果已达到最大的迭代世代，停止；否则，执行第 9 步。

第 9 步：通过杂交和变异操作产生子代，然后回到第 2 步。

8.2.3 数例

Cheng 和 Gen 对一随机产生的测试问题进行了计算测验[109]。这是一个具有 100 个节点和 473 条边的非平面无向图。每条边对应着两个数：时间和费用。进化环境设置如下：种群大小为 40，最大迭代世代数为 1000，杂交概率为 0.4，变异概率为 0.2。

在一次运算中获得如下 Pareto 解：(861,1236),(875,1140),(878,1115),(891,1105),(894,1080),(907,931),(926,904),(927,851),(930,826),(946,824),(949,799),(974,754),(1096,740),(1112,712)和(1243,702)。表 8.1 给出了不同权重的理想点和折中解。表中的理想点是两次运用 Floyd-Warshall 算法，每次对应一个目标得到的。使用的后悔函数分别是 L_1 范数、L_2 范数、L_∞ 范数。图 8.6 中描述了代理理想点的进化过程。

表 8.1 随机测试问题的结果

范数	理想解	$w_1=0.5$ $w_2=0.5$	$w_1=0.2$ $w_2=0.8$	$w_1=0.8$ $w_2=0.2$
L_1 范数	(861,702)	(974,754)	(1112,712)	(930,826)
L_2 范数	(861,702)	(974,754)	(974,754)	(930,826)
L_∞ 范数	(861,702)	(1243,702)	(1243,702)	(861,1236)

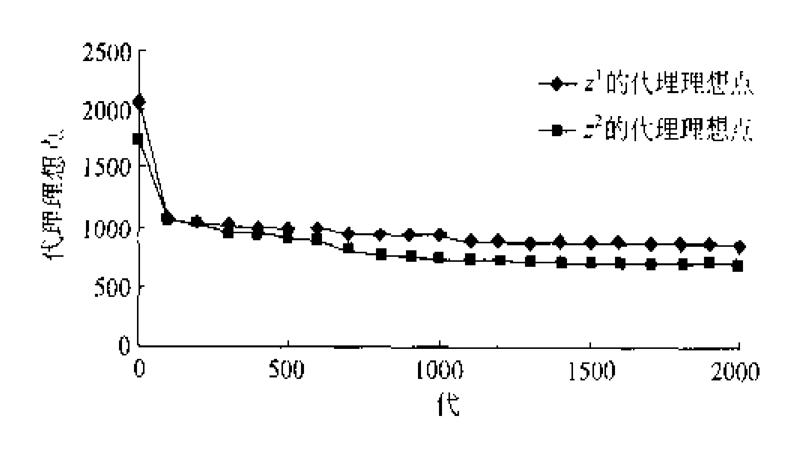


图 8.6 代理理想点的进化过程

8.3 有适应能力的网络路由

将拓扑信息应用于网络，路由算法为通信包确定了一条路径，找到了一条从源节点到目的地的路径。Internet 的迅速发展使得有适应能力的网络路由 (adaptive network routing)的算法更加重要。在早期的 Internet，通常运用基于跳计数矩阵的常规向量距离路由算法。有许多这类算法，包括：路由信息协议(routing information protocol)(RIP)、最短路径优先协议(shortest-path-first protocol)(SPF)以及其他[471]。这些算法的大多数都没有注意到路由中的通信等待时间，仅仅依靠跳计数矩阵寻找最短路径。RIP 在小的局域网(LANs)中至今还常用。这种算法在较大的网络中会产生许多额外的通信，因为它们周期性的发布包含路由表(routing table)的传播消息给网络中所有节点，这会大大降低网络的整体性能。为了减少额外通信，基于链接状态信息交流的路由算法(例 SPF)发布仅包含链接状态信息的传播消息。基于从链接状态信息中产生的拓扑数据库，这种算法在每个节点运用 Dijkstra 最短路径算法计算最短路径。这种算法减少了额外通信，因为它仅传播关于链接状态的信息，而不是在小型网络中的路由表信息。然而，这种算法也有缺陷，即在更大型网络中会产生总体冗余。

基于 SPF 协议，开放最短路径优先协议(open shortest-path-first protocol)(OSPF)是一种广泛应用的网络路径协议，它用作内部网关协议(interior gateway protocol)(IGP)。对于外部网关协议(exterior gateway protocol)(EGP)，BGP4[307]这样的广域网关协议(broader gateway protocol)通常应用于 Internet。BGP4 采用了包含到所有目的地的完整路径的源路径策略，这也不是可标度的。BGP4(或其他的 EGP)将网关分组形成自治区域(AS)，并为每个赋予 AS 标号，限于 0 到 65535 之间，以避免路由表的膨胀。

近年来，有不少关于将遗传算法应用于解决网络路径问题的报道，例如最大流问题，即利用网络的全局信息使流量最大化。Cox[136]提出了遗传算法在网络路径中的另一类应用，解决约束最优化问题，利用有关网络连接带宽的信息为每个连接分配链接带宽。

Carse,Fogarty 和 Munro 将模糊分类系统(FCS)应用于包交换网络中的分布式路由问题。在包交换网络(packet-switching networks)中每个包使用独立路由表进行路由指派。利用 FCS 算法确定包是经直接路径还是间接路径。

Munetomo,Takao 和 Sato 提出了一种适用于 Internet 这样的包交换网络的有适应能力的路由算法。该算法通过观察路由延迟试图最小化通信等待时间。它通常保存了一定数量经常用到的可选路径,并通过在其路由表中可选路径间分配包,试图平衡链上负载。Munetomo,Takao 和 Sato 应用遗传算法构建路由表。路由表就是一群串,每个串就代表一条路径。

8.3.1 基于遗传算法的有适应能力的路由

基于遗传算法的有适应能力的路由(genetic-based adaptive routing)(GAR)算法应用遗传路径算子在其路由表中创建可选的路径。基于源路径的方法,GAR 算法有效地利用路由表中路径间的相似性。源路径算法为源节点上的包确定了一条完整的路径,因此一个包需要包含路径上所有节点的信息。Internet 协议拥有说明源路径是否使用源路径的选择权[66,132]。数据包通常不使用这项权利,仅仅通过查询基于网络最短路径的路由表来确定下一跳。利用 GAR 算法生产的路径中创建仅描述下一跳的路由表并不明显也不困难。GAR 算法的重要特征总结如下:

- GAR 算法是源路径算法,在路由表中的每一条路径包含了路径上的所有节点。
- 对于包的每一个目的节点,都存在一组可选路径。
- 每一条路径都有它的权重值,它详细说明了这条路径在可选路径中被选择的概率,且对应着遗传运算的一个适合值。
- 利用发送延迟查询包观测得到的通信等待时间计算路径的权重。
- 利用基于跳计数准则的 Dijkstra 最短路径算法产生默认路径,作为初始路径。
- 路由表中的可选路径是通过变异、杂交之类的遗传运算产生的,这些路径按每次权重计算后的概率被调用。
- 为了避免路由表的溢出,它的大小通过应用选择运算被减小。到达同一目的地的路径中权重最小的路径被删除。而且当标号最小的包被送到路由表中的目的地时,到达这一节点的所有路径被删除。

8.3.2 染色体表示

基于网络的拓扑数据库,路径是通过列举从源点到其目的地的节点来编码。例如,一条从节点 0 到节点 9 的路径可以编码成沿路径的一串节点:(0 12 5 8 2 9)。如果路径在网络上不能实现,则它就不能被编译成一染色体,这意味着路径中的每一步都必须经过网络中实质上的连接。

表8.2是在路由算法中使用的路由表。表格由5个实体组成：目的地、路径、频数、延迟和权重。目的地指明包的目的节点，对应每个目的地有一组可选路径，路由记录(routing entry)是沿路径的一串节点，频数描述了由路径送往目的地的包的数量，延迟显示了沿路径发送包的通信等待时间，权重描述了包发送时该路径被选择的概率。

表8.2 路由表例子

目的地	路 径	频 数	延 迟	权 重
2	(1 3 2)*	7 232	50	0.7
	(1 3 4 2)	2 254	60	0.2
	(1 3 4 5 2)	1 039	70	0.1
6	(1 8 6)*	20 983	100	0.4
	(1 10 11 6)	34 981	105	0.6
8	(1 8)*	30 452	40	0.9
	(1 7 8)	3 083	40	0.1

表8.2中标有星号的路径是默认路径，根据跳计数准则的最短路径。在初始状态，路由表是空的。当在某一节点产生包时，利用Dijkstra算法产生一条默认最短路径，插入路由表。沿路径发送指定数量的包后，发送一个包用于计算该路径通信等待时间。接受到包的回答后，路径遗传运算(path genetic operators)以指定的概率应用于全部可选路径。

8.3.3 染色体评价

路径权重值(适应值)评价是基于沿路径的通信等待时间。定时发送延迟查询信息用于观测沿该路径的延时。利用得到的延迟值，路径权重值的计算如下：

$$\text{eval}(v_k)=\frac{1/D_k}{\sum_{i\in S}1/D_i} \tag{8.23}$$

其中 $D_i=d_i\Big/\sum_{j\in S}d_j$；$d_i$ 是路径 i 的延迟，S 是同一目的地的路径集合。等式(8.23)表明 d_i 值越小，产生的评价值越大。通过评价，具有较短通信等待时间的路径被频繁地用于传输包。

8.3.4 遗传算子

路径杂交(path crossover)运算是交换两个染色体中的子路径。染色体必须拥有相同的源节点和目的节点。路径杂交运算的杂交位置限制在两个染色体中都含有的节点。从潜在的杂交位置中随机选择节点作为杂交位置，交换子路径。当杂交运算应用于一对染色体 V_1 和 V_2 时，操作过程如下。

过程 路径杂交运算

第 1 步：列举 V_1 和 V_2 中均含有的节点集合 N_c（不包含源节点和目的节点）作为潜在的杂交位置。

第 2 步：从 N_c 中选择一个节点 i 作为杂交点。

第 3 步：通过交换杂交点后的所有节点得到杂交的染色体。

图 8.7 显示了将杂交运算应用于从节点 0 到节点 20 的一对父代 V_1 和 V_2 的概况。它们潜在的杂交位置是节点 7,11 和 15。当我们选择节点 11 作为杂交位置时，如图 8.7 所示通过交换子路径产生新的子代。

当一对染色体中的公共点不存在，则杂交位置就不能选择，因此也就不可能实施杂交运算。没有相似性的一对父代是不值得杂交的，因为杂交可能会产生远离父代的路径，这意味着路径的随机再现。

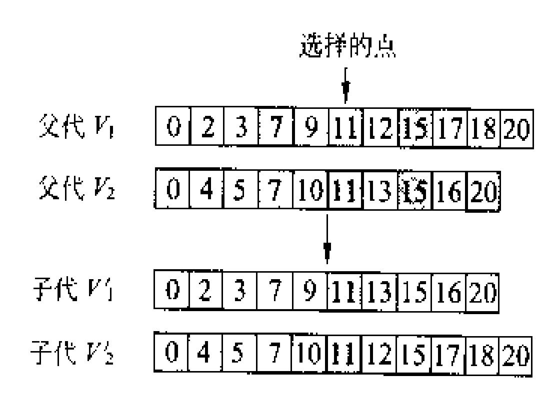


图 8.7 杂交运算

路径变异(path mutation)运算是从一染色体产生另一染色体。为了实现变异，从染色体中随机选择节点。这个节点称为变异节点。然后，从与变异点直接相连的节点中随机选择另一个节点。最后，根据 Dijkstra 最短路径算法，通过连接源节点到选择点和选择点到目的节点产生可选路径。路径变异运算描述如下。

过程 路径变异运算

第 1 步：从父代 V 的所有节点中随机选择变异节点 i。

第 2 步：从变异节点 i 的邻点中选择节点 j，$j\in B$，B 是变异节点 i 的邻点集。

第 3 步：产生从源节点到 j 的最短路径 r_1 和从 j 到目的节点的最短路径 r_2。

第 4 步：如果 r_1 和 r_2 中存在重复节点，则放弃路径，不进行变异。否则，连接两段路径组成变异后的染色体。

图 8.8 显示了变异运算的一个例子。首先，假设节点 7 被选择作为变异点。其次，从变异节点的邻点中选择节点 8。再次，根据 Dijkstra 最短路径算法，我们连接源节点 0 到节点 8 产生子路径 r_1，连接节点 8 到目的节点 15 产生子路径 r_2。连接 r_1 和 r_2，完成变异操作。如果子路径 r_1、r_2 中存在重复节点，子代 V' 就不能产生，因为我们必须避免路径中的任何环。

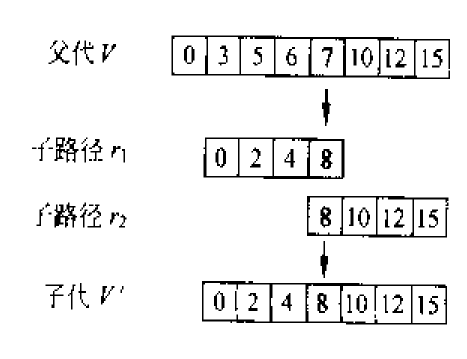


图 8.8 变异运算

GAR 算法描述：Munetomo，Takai 和 Sato 的路由算法详情描述如下[474]。每个节点独立执行这一算法以确定包的路径。每个包具有类型、路径和下一跳三个实体，其中类型表示包的类型，路径是包

的传输路径，下一跳指明了在它的路径中包的下一跳。包的类型有：数据包(data packet)(包含数据的包)、延迟请求(delay request)(路径中请求延迟的包)和延迟回答(delay answer)(回答延迟请求的包)。延迟请求和延迟回答有延迟记录(delay entry)，它们记录包的通信等待时间。

如果到达某一节点的包的类型是数据包，则节点根据其路由表向前传递包。如果包是在某一节点产生，则根据节点路由表确定包的路径。如果路由表中不存在达到输入包目的节点的路径，则利用 Dijkstra 最短路径算法产生一条默认路径插入路由表。在算法的初始状态，路由表中不存在任何路径。该算法仅产生那些通往频繁有通信包发送到的目的节点的路径。为了观测路径的通信等待时间，延迟请求以特定的间隔沿路径发送。如果包达到目的地，则延迟回答包被往回发送。当延迟回答包收到时，通过计算从发送延迟请求包到收到延迟回答包的平均时间，便得到路径的通信等待时间。

GAR 算法

```
begin
    初始化路由表；
    While 终止条件不满足 do
      begin
        等待接受或输入来自用户的包；
        if 包类型=数据包 then
        begin
            if 包发自于其他节点 then
            begin
                if 包目的地=该节点 then
                    接受包；
          else
              将包送往下个包的节点；
            end
            else
            begin
                if 目的地路由表是空集 then
                begin
                    由 Dijkstra's 算法创建缺省路由；
                    添加缺省路由到路由表；
                    重新设置缺省路由的频率记录；
                    包路由←缺省路由；
                    if 目的地数 >限值 then
                        删除一目的地使用最少的路由；
                end
```

```
            else
            begin
                用轮盘赌选择从路由表中选择一路由；
                增加选中路由的频率记录；
                包路由← 选中路由；
                if 路由频率 mod 评价间隔=0 then
                begin
                    路由类型 ← 延迟请求；
                    包路由← 路由；
                    按路由发送包；
                end
            end
        end
    end
    if 包类型=延迟请求 then
    begin
        包延迟记录 ←当前时刻一包创建时刻；
        包创建时刻←当前时刻；
        包类型←延迟回答；
        将包送往源点；
    end
    if 包类型 ←延迟回答 then
    begin
        包延迟记录←包延迟记录+当前记录一包创建时刻；
        包延迟记录←包延迟记录/2；
        改变基于包延迟记录路由的延迟；
        if 随机数< p_M then
        begin
            对路由表中某染色体应用变异操作；
            将产生的串添加到路由表；
            if 表的大小>限值 then
                删除权重最小的串；
            end
            if 随机数< p_C then
            begin
                对路由表中某一对染色体应用杂交操作；
                将产生的串添加到路由表；
                if 表的大小>限值 then
                    删除权重最小的串；
```

```
            end
         end
      end
   end
```

接收到路径延迟后，根据式(8.23)计算同一目的地的路径的权重值。每次计算权重值后，按特定的概率应用遗传算子创建路由表中的可选路径。如果路由表的大小超过限制，则执行选择操作减少它的大小。有两类选择操作：

1. 局部选择：删除同一目的地的路径中权重最小的路径。
2. 全局选择：删除在路由表中到达所有目的地节点中，使用频率最小的目的地路径。

8.3.5 数例

使用基于离散事件模拟的网络模拟器，Munetomo，Takai 和 Sato 进行了一些试验[474]。他们比较了 GAR 与 RIP，SPF 之类的常规算法。图 8.9 中描述了一个网络例子，该网络是根据日本主要城市的有关地理信息绘制的。线的宽度表示链路的带宽。较宽的是 4.5Mbps，较窄的是 1.5Mbps。包是按特定的呈指数分布的间隔随机产生的。包的目的地是从用阴影圆圈表示的节点中随机选择的。

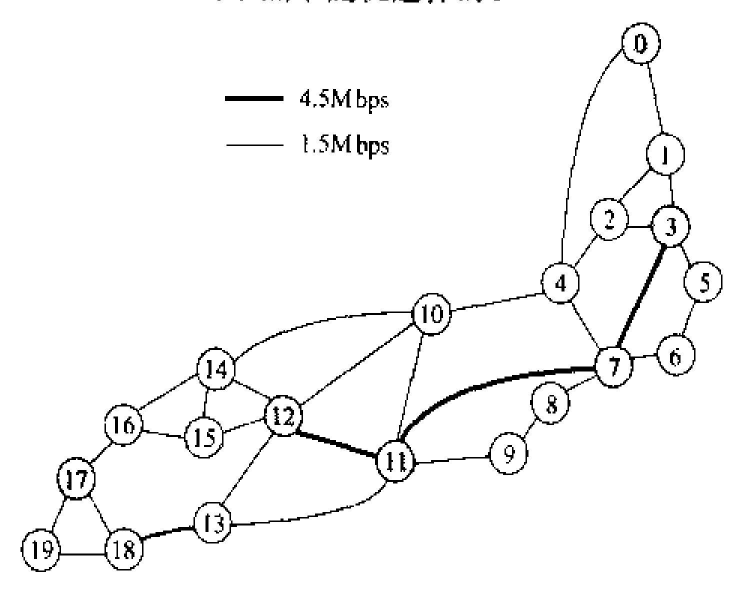


图 8.9 网络例子

模拟环境如下：每隔 10 个包计算一次染色体(路径)的适应值。应用变异的概率 $p_m=0.1$，应用杂交的概率 $p_c=0.05$。种群大小为 100。模拟进行了 3000s，模拟后得到如下结果。为比较数据包的平均响应时间和它们在源节点产生到抵达目的地的平均时间，可以通过改变呈指数分布的数据包平均产生时间间隔，来改变数据包产生的频率。

从结果中，我们可以看到应用 GAR 算法得到包的平均响应时间最短，SPF 稍优于 RIP。尤其对于 2000ms 的数据包产生间隔，这将导致网络中链路的重负荷。GAR 得到

的平均响应时间是 RIP 和 SPF 的 20%。这意味着利用 GAR 发送的包到达目的地的速度比其他算法快 5 倍。

表 8.3 试验后节点 0 的路由表

目的地	路 径	延迟	权 重
3	(0 1 3)	554	0.802636
	(0 4 2 3)	2253	0.197364
7	(0 4 7)	3052	1.000000
11	(0 4 7 11)	4423	0.533488
	(0 1 3 7 11)	5058	0.466512
12	(0 4 7 11 12)	2941	0.564116
	(0 4 10 12)	6210	0.267160
	(0 1 2 4 10 12)	9833	0.168724
17	(0 4 10 14 16 17)	2859	1.000000

表 8.3 显示了一次包产生时间间隔为 2200ms 的模拟后，节点 0 上的路由表。对于目的节点 12，最优路径是(0 4 7 11 12)。在跳计数准则下，另一路径(0 4 10 12)是最短路径，但不是延迟最小的路径，因为从节点 7 到 11 和从节点 11 到 12 的链路带宽比其他路径宽。这一结果也表明了可选路径间的负载平衡是通过概率地根据它们的权重值分发包实现的。图 8.10 至图 8.12 显示了模拟期间链路的负载状况。每条链路的宽度代表链路的对数值的平均排队长度。因为宽度是取对数的，所以粗线意味着大大超载。在 RIP 的结果中，路径(11 13 18)极大超载，另一方面，(11 13 18)的可替代路径(11 12 15 16 17 18)

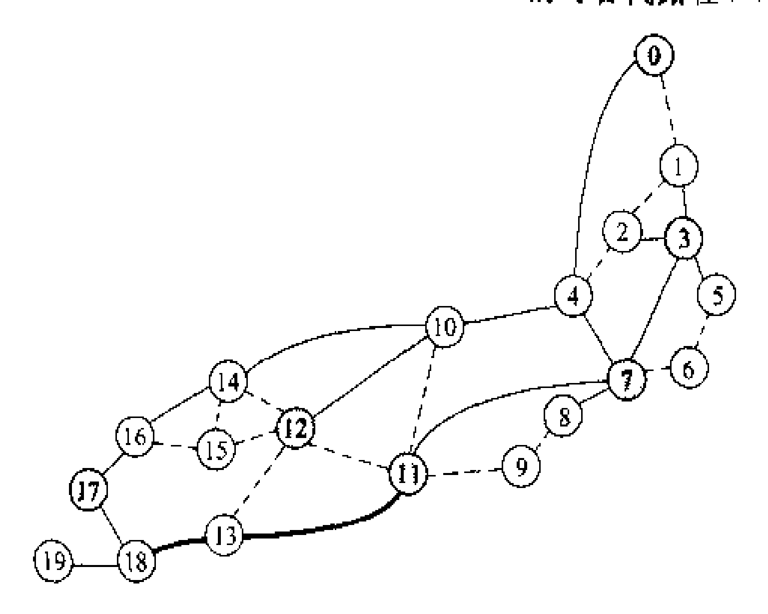


图 8.10 链路负载状态(RIP)

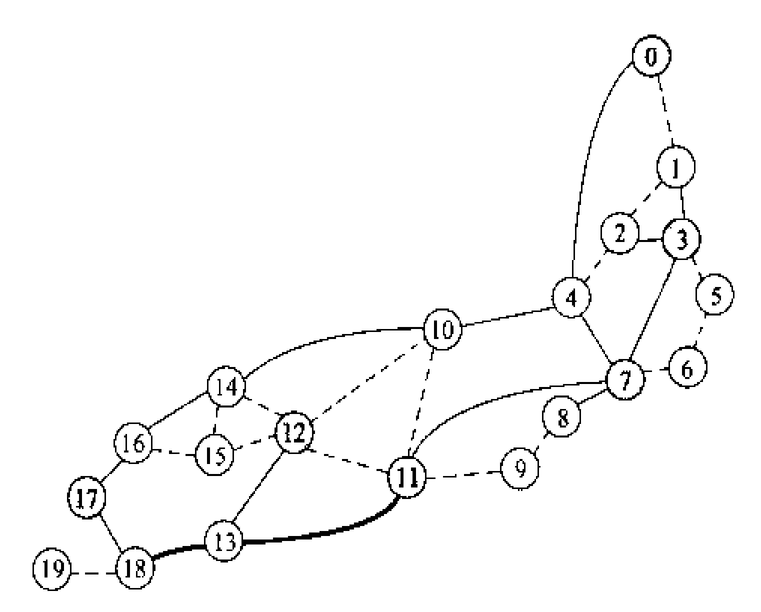


图 8.11　链路负载状态(SPF)

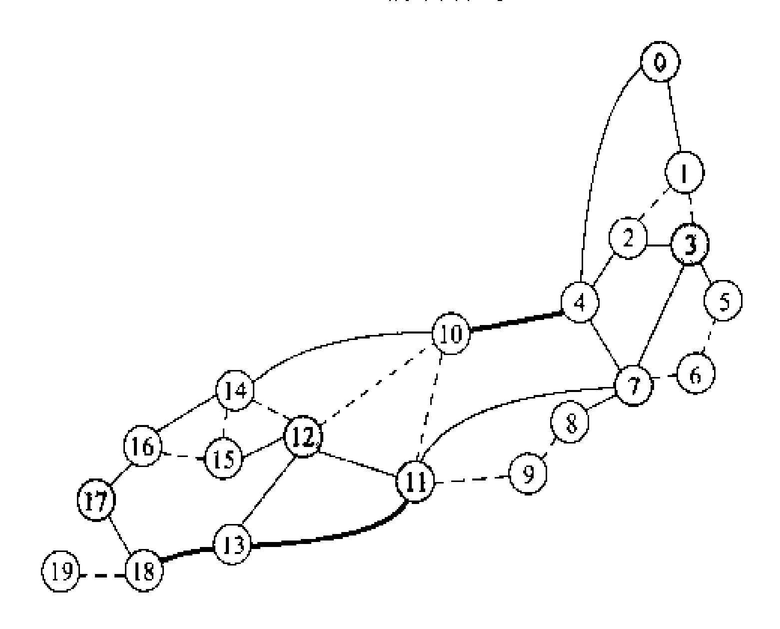


图 8.12　链路负载状态(GAR)

很少使用，负载很小。应用 SPF，链路的负载有所减少，但本质上与 RIP 一样。另一方面，应用 GAR 算法，链路的负载，尤其是重负荷的链路上的负载能够大大减少。这不仅仅因为 GAR 算法计算的路径通信等待时间最小，还因为其在路由表中的可选路径中分发包，这可以减轻重载链路的负荷。以上结果表明了包的平均响应时间可以更缩短，尤其在重载的网络中，原因如下：

1. 考虑路径通信延迟。
2. 应用遗传算子产生可选路径。
3. 在可选路径中分发包，以实现路径间的平衡负载。

8.4 集中式网络设计

集中式网络(centralized network)是一个所有通信都来往于一个点的网络[357]。在这样的网络中,终端被直接连接到中央节点(central site)。有时候还使用多点线,此时一组终端共享在一棵树上连接到中央节点,且每条多点线仅通过一条链路连接到中央节点。这意味着这一问题的最优拓扑结构对应图 $G(V,E)$ 中的一棵树,V 中除一个节点外所有节点都对应着一个终端,剩下的那个点为中央节点,E 中的边对应可远程通信的线路。每一棵根植于中央节点的子树对应一条多点线。通常在通信中,中央节点最多可以处理给定固定量的信息。反过来,这也相应地限制了任何连接到中央节点(称其为图 G 的根)的链路上的信息传输最大量。在组合优化的文献中,这一问题就是著名的容量限制最小生成树问题(capacitated minimum spanning tree problem)。

Papadimitriou 认为这是 NP-难题[502]。许多早期工作利用启发式方法寻找良好的可行解。Chandy 和 Lo[94],Kershenbaum[356],Elias 和 Ferguson[175]对此做了很多工作。我们目前所知道的仅有的全部优化算法由 Gavish[210]和 Kershenbaum 提出。但他们的算法仅限于不多于 20 个节点的问题。对于容量限制最小有向树问题,Gavish[211]也研究了一套新的构造及其一些松弛程序。最近,这一问题已引起了人们对 Gouveia[261]和 Hall[277]的切平面法,以及 Malik 和 Yu[434]的分支定界法的更大兴趣。Zhou 和 Gen 提出了遗传算法的方法用于处理这类问题[696,699],一种基于树的编码方法被用于候选解的编码。

8.4.1 问题的描述

考虑完全无向图 $G=(V,E)$,令 $V=\{1,2,\cdots,n\}$ 是代表终端的节点集合。表示中央节点或"根"的节点为节点 1。令 $E=\{(i,j)\mid i,j\in V\}$ 是代表所有可能远程通信线路的边集合。对于节点子集 $S(\subseteq V)$,定义 $E(S)=\{(i,j)\mid i,j\in S\}$ 为两端点都在 S 中的边集。对于所有的边 $(i,j)\in E$,定义下面的二元决策变量:

$$x_{ij}=\begin{cases}1, & \text{如果边}(i,j)\text{被选中}\\0, & \text{其他}\end{cases}$$

令 c_{ij} 是解中边 (i,j) 的固定成本。假设 d_i 代表节点 $i\in V$ 的需求量,其中约定根节点的需求量 $d_1=0$。$d(S)$,$S\subseteq V$ 代表 S 中节点需求量之和;子树的容量用 κ 表示。

集中式网络设计(centralized network design)问题可以公式描述如下[310]:

$$\min \quad z(x)=\sum_{i=1}^{n-1}\sum_{j=2}^{n}c_{ij}x_{ij} \tag{8.24}$$

$$\text{s.t.}\quad \sum_{i=1}^{n-1}\sum_{j=2}^{n} x_{ij} = 2(n-1) \tag{8.25}$$

$$\sum_{i\in S}\sum_{\substack{j\in S\\ j>1}} x_{ij} \leqslant 2(|S| - \lambda(S)),\quad S\subseteq V\backslash\{1\},\quad |S|\geqslant 2 \tag{8.26}$$

$$\sum_{i\in U}\sum_{\substack{j\in U\\ j>1}} x_{ij} \leqslant 2(|U| - 1),\quad U\subset V,\quad |U|\geqslant 2,\quad \{1\}\in U \tag{8.27}$$

$$x_{ij} = 0 \text{ 或 } 1,\quad i = 1,2,\cdots,n-1,\quad j = 2,3,\cdots,n \tag{8.28}$$

式(8.25)对于所有的生成树恒为真：一棵 n 个节点的树必定有 $n-1$ 条边。不等式(8.27)对于生成树是一个标准不等式。如果多于 $|U|-1$ 条边连接子集 U 中的节点，则子集 U 中必包含圈。不等式(8.26)中的 $\lambda(S)$ 是指集合 S 的装箱数，即用大小为 κ 的箱子装项目大小为 d_i 的节点所需要的箱子数目，所有的 $i\in S$。这些约束类似于不等式(8.27)，当然，它们并不代表容量限制。如果集合 S 不包含根节点，则 S 的节点必须包含在至少 $\lambda(S)$ 棵不同的子根树中。

当所有非根节点的需求量为1时，不等式(8.26)可以简化表示为

$$\sum_{i\in S}\sum_{\substack{j\in S\\ j>1}} x_{ij} \leqslant |S| - \left\lceil \frac{|S|}{\kappa} \right\rceil,\quad S\subseteq V\backslash\{1\},\quad |S|\geqslant 2 \tag{8.29}$$

因为单位大小的对象永远可以被装入 $\lceil |S|/\kappa \rceil$ 个箱子或子树中。

目前，所有这类问题的启发式算法仅集中于如何处理约束，以使问题更易于求解。在割平面算法[261,277]和分支定界算法[434]中，问题的拓扑结构往往被忽略。因此，这会导致约束个数的指数膨胀。

8.4.2 遗传算法

为了利用遗传算法解决集中式网络设计问题，候选解的编码采用了基于树的排列(tree-based permutation)，图8.13中给予了说明。

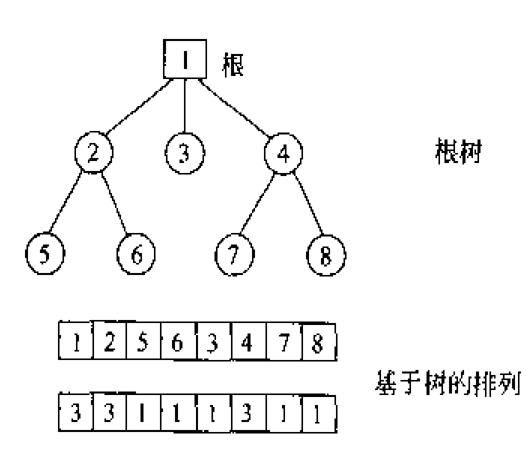


图8.13　根树和基于树的排列

特别地，对于根节点，它的度不应小于 $\left\lceil \frac{|V|}{\kappa} \right\rceil$。这反映了连接到根节点的子树的数量满足容量限制。对于初始种群，除了节点1永远是节点维的第一个基因，每条染色体都是随机产生的。

这里使用的遗传算法与2.5节中用于解决度约束的最小生成树问题的遗传算法相同。但是，对进化过程中产生的超出根节点容量的不可行解进行修正是必要的。尤其当所有终端的需求量等于1时，问题可简化为寻找每棵根节

点的子树最多包含 κ 节点的生成树。因此，计算前，如果存在其子树超出容量限制的个体，再次进行变异运算将多余的树枝插入另一棵具有较少节点的树。进化过程描述如下：

第 1 步：根据 2.5 节中基于度的排列编码方法的编码过程，将一个体转化为对应的树解。

第 2 步：根据问题中的目标函数(8.24)计算解的总成本，将总成本的倒数作为适应值。

第 3 步：对所有个体重复过程。

在此，采用($\mu+\lambda$)选择策略。为了避免进化过程的过早收敛，选择策略仅从 μ 个父代和 λ 个子代中选择 μ 个不同的最佳个体。如果不存在 μ 个可选个体，则种群池中的空缺填入按初始个体同样方式产生的新个体。

8.4.3　数例

在 Gavish[212] 给出的实例上，Zhou 和 Gen 测试了他们的方法。该例子包含 16 个节点，每个节点与节点 1 之间为单位通信量，容量限制 $\kappa=5$。成本矩阵如表 8.4。

表 8.4　数例的成本矩阵($n=16,\kappa=5$)

i/j	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1616	1909	246	622	829	1006	2237	399	1717	632	1191	2116	824	1336	1519
2		2996	1419	2217	1213	2046	3753	1516	1180	1997	552	3622	2423	1367	862
3			1893	1543	1792	2785	1362	1667	3556	2332	2446	1248	1508	3233	3287
4				799	593	1185	2369	242	1670	857	962	2243	1004	1348	1425
5					1230	1253	1625	761	2301	801	1748	1509	206	1873	2119
6						1758	2597	480	1883	1449	663	2463	1420	1701	1573
7							2703	1399	1470	454	1849	2612	1350	960	1470
8								2238	3922	2304	3231	137	1442	3476	3743
9									1889	1029	1009	2108	959	1586	1628
10										1693	1437	3808	2480	511	340
11											1685	2206	909	1205	1603
12												3098	1952	1429	1100
13													1331	3368	3624
14														2038	2309
15															578

遗传算法的参数如下设置：种群大小 200，变异概率 $p_m=0.3$，最大繁殖世代数为 500。Gavish 使用扩展拉格朗日算法解决这一问题，得到最优解 8526[212]。使用遗传算法得到同样的最优解。它相应的树形拓扑结构如图 8.14 所示。

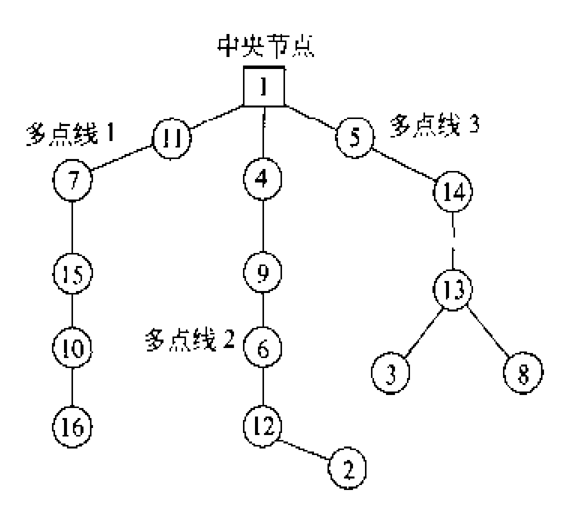


图 8.14　问题的最优解(成本为 8526)

8.5　计算机网络扩展

低价计算机设备的出现导致了计算机网络的爆炸性发展。通过网络的分布式计算带来了不少好处。计算机网络较之集中式系统的主要优势之一是其提高可靠性的潜力。系统的可靠性不仅依靠它的节点和通信线路的可靠性,还依赖于节点是如何通过通信线路被连接的(例如,网络的拓扑结构)。网络拓扑结构可由无向图 $G=(V,E)$及其网络的可靠性、信息延迟和网络容量来刻画,其中,N 是节点集,E 是边集。这些性能特性依赖于图的许多属性,例如:直径、平均距离、每个节点的端口数(节点的度)和边的数量。直径越大,网络中的延迟越长;可靠性随直径和平均距离的减少而增加,随边数的减少而降低。

网络扩展(network expansion)是针对更多跨网络计算而带来的不断增长的需求。为满足这一需求,网络的大小随着用户需求的增加不断地膨胀。在这样一个例子中,扩展是通过适当地增加新的通信线路和计算机节点实现的,以至于网络的可靠性评价在特定的限制内得到优化。

8.5.1　问题描述

Kumar、Pathak 和 Gupta 开发了基于遗传算法的计算机网络扩展(computer network expansion)方法,用于在给定的一组网络约束下,优化特定的目标函数(可靠性评价)[379,380]。对于网络扩展问题,定义以下假设:网络的拓扑结构由无向图 G 描述,它的节点代表处理单元,边代表通信线路。图 G 中不存在任何自环。任何边的故障统计意义上是相互独立的。边有2个状态:运行和故障。网络是统计上耦合的。

在直径和度约束条件下,最大化计算机网络可靠性(network reliability)的网络扩展问题可以构造如下:

$$\begin{aligned}\min\quad & R(\boldsymbol{x})\\ \text{s.t.}\quad & \sum_{j=1}^{n} x_{ij} \leqslant k_i, \quad i=1,\cdots,n\\ & \max\{d_{ij} \mid i=1,\cdots,n;\quad j=1,\cdots,n\} \leqslant D\end{aligned}$$

其中 n 是节点数，k_i 表示节点 i 的度。如果节点 i 和 j 间存在边，则 $x_{ij}=1$；否则 $x_{ij}=0$。d_{ij} 是节点 i 和 j 间跳跃的最小数。D 是网络直径的上界。

8.5.2 Kumar，Pathak 和 Gupta 的方法

染色体表示和初始化 为了表示节点的连通性，可以使用一种二进制串表示的编码方法。网络连通性的完整染色体被分成节点域。节点域的多少等于新的节点加入现有网络后网络中的节点数。图 8.15 给出了三个节点网络连通性的一种排列，其中，一个新节点（x'_3）加入到现有的两节点（x_1 和 x_2）网络。一般来说，如果在扩展的网络中有 n 个节点，开始的 $i-1$ 个节点表示原先网络中的节点，则剩下的节点是新增入网络的。这一扩展网络的染色体如图 8.16 所示。通常 x'_{nn} 和 x_{nn} 为 0，因为图中不允许存在自回路。初始种群完全是随机产生的，在初始化中不使用任何准则产生有偏的或有适应能力的种群。

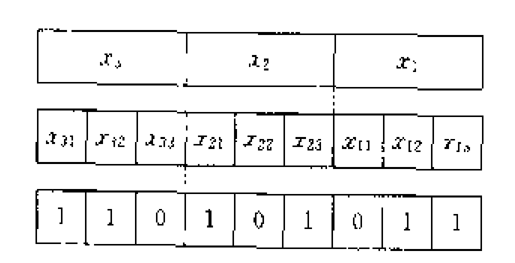


图 8.15 网络扩展问题的染色体表示

x_n	$\cdots$	x_i	x_{i-1}	$\cdots$	x_1

x_n	x_{n1}	$\cdots$	$x_{n,i-1}$	x_{ni}	$\cdots$	x_{nn}
x_i	x_{i1}	$\cdots$	$x_{i,i-1}$	x_{ii}	$\cdots$	x_{in}
x_{i-1}	$x_{i-1,1}$	$\cdots$	$x_{i-1,i-1}$	$x_{i-1,i}$	$\cdots$	$x_{i-1,n}$
x_1	x_{11}	$\cdots$	$x_{1,i-1}$	x_{1i}	$\cdots$	x_{1n}

图 8.16 网络扩展染色体的一般表示

杂交 遗传杂交运算分为两类：(1)SNFSO（简单节点域交换运算），(2)RNFSO（随机节点域交换运算）。在 SNFSO 中，杂交发生在从前一代中随机选出的两条染色体的节点域边界上。然后，利用随机数发生器在 $(0,n-1)$ 上产生一整数作为杂交点。杂交产生两条带有父代信息的新的染色体。图 8.17 显示了当杂交点在第二个决策变量 x_2 右边界时，杂交运算的操作过程。

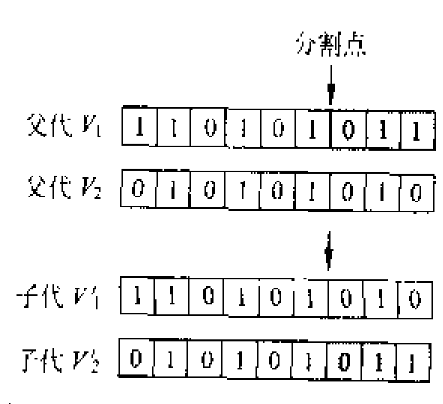


图 8.17 简单节点域交换运算(SNFSO)

有时候杂交运算会产生网络连通性不合理的染色体。例如：子代 1 中的节点域 x_3 显示了节点 3 连接节点 1 和 2，但 x_1 表明节点 1 仅连

接节点2。这种异常可以通过后面要讨论的调整运算改进。

在RNFSO中，每次随机交换染色体中的节点域。为了交换一特定标号的节点域，随机数将在[0,$n-1$]中产生。例如：如果仅有一个节点域要交换，则产生一个随机数来决定哪一个域要交换。假设，要交换的域是3，则RNFSO过程可由图8.18说明。正如SNFSO所讨论的，RNFSO也会产生异常的染色体，同样可以通过调整运算改进。对于网络的扩展问题，RNFSO是首选，因为它为每个节点连接自己与其他节点提供了无偏的相等机会，并通过调整运算保证了它们的连通性。但使用SNFSO会使得低标号的节点占据优势，通过调整运算迫使其他节点改变它们的连通性。

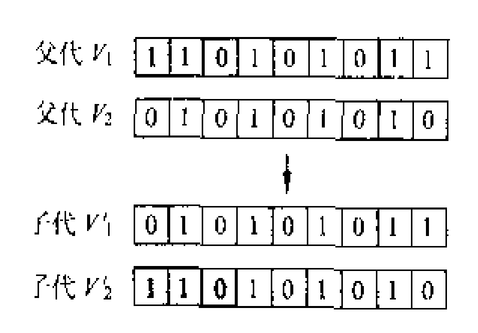


图8.18　随机节点域交换运算(RNFSO)

调整运算　由于这些杂交运算生产了染色体连通性上的矛盾，需要执行下面的简单过程：

过程　交换节点域(x_i)的调整运算

第1步：令 $j=1$。

第2步：如果 $x_{ij}=1$，则执行第3步；否则，执行第4步。

第3步：如果 $x_{ji}=0$，则令 $x_{ji}=1$。

第4步：如果 $x_{ji}=1$，则令 $x_{ji}=0$。

第5步：更新 j，即 $j=j+1$。如果 $j>n$，则停止；否则，执行第2步。

根据新需要的域对染色体进行调整，这种调整运算保持了杂交的特性。

突变　如果说杂交运算稍稍降低了节点域的连通性，则变异运算将提高节点域的连通性。以下是运算过程中的4个步骤：

第1步：从那些加入现有网络的，且度数小于上界的点中，随机选择节点域 x_i。

第2步：随机选择另一个不与 x_i 相连的，且度数小于上界的节点 x_k。

第3步：如果不存在这样的点，则执行第1步。如果进行变异的节点 x_i 的个数超过变异的次数，则停止。

第4步：如果找到了合适的节点 x_i 和 x_k，通过令节点域 x_i 中的 x_{ik} 位为1，令节点域 x_k 中的 x_{ki} 位为1，产生一条新的边。如果到此为止所进行的变异次数小于 $p_M \cdot pop_size \cdot str_len$ 确定的次数(其中 str_len 是染色体的长度)，则执行第1步；否则，停止。

选择和终止规则　Kumar，Pathak和Gupta的方法中，如果每次杂交、变异和调整后，子代比父代具有更好的目标函数值，则加入新的迭代世代。如果子代仅优于一个父代，则子代和较优的父代中随机选择一个。如果子代比两个父代都劣，则从两个父代中随机选择一个作为下一代。这确保了永远是最优的染色体进入下一世代，而劣的不进下一世代。当染色体的平均适应值和最大适应值一致时，遗传运算终止。

适应值和可靠性的计算 既然目标函数是可靠性函数，则适应值由可靠性的计算决定。计算机网络的可靠性(computer network reliability)定义为在给定网络中，每个节点与其他所有节点通信的概率。网络可靠性的计算，可使用以下过程：

第 1 步：利用 Aggarwal 和 Rai 的算法[8]，计算网络中所有的生成树，然后将其分隔开来。图 G 表示网络的生成树 T_i 被定义为 G 的连通子图，它包含 G 中的所有节点。根据生成树的定义，任何 T_i 通过 $n-1$ 条边连接 G 中所有的点，因此是满足节点间通信的最小连接。

第 2 步：利用预先给出的边的可靠性，计算被操作的生成树的可靠性，进而用于网络可靠性的计算。

整个算法

第 1 步：设置参数。设置种群大小(pop_size)，杂交概率(p_c)，变异概率(p_m)，最大繁殖世代数(max_gen)和初始繁殖世代数 $gen=0$。同时也要输入节点数、节点的度、边的可靠性和网络直径的上界(D)。

第 2 步：随机产生初始种群。

第 3 步：评价、计算当前世代中每条染色体的网络可靠性。

第 4 步：从当前世代中选择染色体，形成配对池。

第 5 步：执行杂交和变异

(5.1) 在配对池中应用 SNFSO 或 RNFSO。

(5.2) 对新的子代进行调整。

(5.3) 如果约束方程不满足，则放弃染色体，执行第 6 步。

(5.4) 应用变异运算。

(5.5) 如果 $n<pop_size-1$，则执行第 6 步，n 为新的子代的个数。

(5.6) 如果适应值优于父代，加入新的迭代世代；否则，放弃染色体。

(5.7) 执行第 6 步。

第 6 步：

(6.1) 令 $gen=gen+1$。

(6.2) 如果终止条件不满足，返回第 3 步，进行下一代繁殖；否则，终止。

8.5.3 数例

Kumar，Pathak 和 Gupta 测试了 4 个例子[380]。在满足网络直径和度的约束下，添加节点直至网络的可靠性极大化。每个例子的约束条件见表 8.5。表 8.6 给出了结果，在所有的例中利用遗传算法得到最优解。

表 8.5　测试问题的参数

节点数	k_i	$\max\{d_{ij}\}$
4+2	4	3
5+2	3	3
6+2	3	3
6+3	$3(i=1,\cdots,6)$	4
	$2(i=7,8,9)$	

表 8.6　测试问题的结果

节点	$R(x)$
4+2	0.9993
5+2	0.9814
6+2	0.9906
6+3	0.9579

8.6　多阶段工序计划

多阶段工序计划(multi-stage process planning,MPP)问题在制造系统中很常见。一般而言,该问题对通过多个工序阶段将原材料转化为成品的生产能力和需求提供了详细的描述。在现代制造系统中,MPP 可分为两类:派生 MPP 和生成 MPP。在派生 MPP(variant MPP)中,将具有相似制造需求的一族部件分在一起并使用编码策略进行编码。当要制造新的部件时,它的代码映射到这些部件族中之一,并获得相应的工序计划。在生成 MPP(generative MPP)中,根据一些决策逻辑,新的工序计划是自动生成的。决策逻辑与某些自动化制造特性识别机制相互作用,通过优化技术得到相应的机器加工要求,因为生成 MPP 包含了一般的自动工序计划模型,这在柔性制造系统中是非常重要的,所以生成 MPP 问题现在特别受到关注[387,731]。

生成 MPP 问题自行引入优化设计技术,为了在给定的一个目标,例如成本最小、时间最小、质量最好,或是多个目标下,从大量可选计划中寻找最佳工序计划。所有这些可选计划的隐枚举可以如网络流一样描述,但是,随着问题规模的增大,难度会如“维数爆炸”一样。甚至使用著名的最短路径算法,例如,Dijkstra [164],Floyd[326] 或 out-of-killer[666],处理更大规模的 MPP 问题也不太容易。特别是,对于一个多目标的 MPP 问题,传统的技术无法使用,尽管有人运用了基于目标规划的技术,但这些技术只能处理小型网络[illegible]。Zhou 和 Gen 提出了用于处理 MPP 问题的遗传算法[illegible]。

8.6.1　问题的描述

MPP 系统通常包含一系列的机械操作,诸如削、钻、磨、精加工等,将一零件加工成最终形态或产品。整个过程可以分为几个阶段,在每个阶段,都有一套相似的加工操作。MPP 问题就是要在给定的目标,如成本最小、时间最小、质量最好,或是这些部分目标下,从所有可能的可选计划中,寻找最优的工序加工计划。图 8.19 显示了一个简单的 MPP 问题。

对于一个 n 阶段的 MPP 问题，令 s_k 是阶段 k 的一种状态，$D_k(s_k)$ 是在阶段 k 可能的状态集合，$k=1,2,\cdots,n$，x_k 是阶段 k 确定选择某种状态的决策变量，显然 $x_k\in D_k(s_k)$，$k=1,2,\cdots,n$。然后，MPP 问题可以构造为

$$\min_{\substack{x_k\in D_k(s_k)\\ k=1,2,\cdots,n}} V(x_1,x_2,\cdots,x_n)=\sum_{k=1}^{n} v_k(s_k,x_k) \tag{8.30}$$

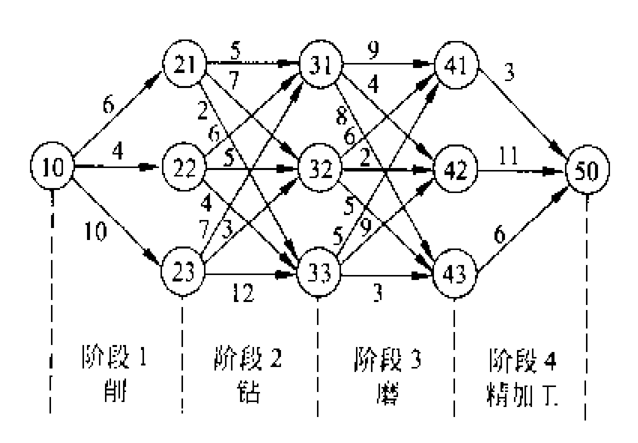


图 8.19 一个简单的 MPP 问题的流程网络

其中，$v_k(s_k,x_k)$ 代表在阶段 k 状态 s_k 中确定 x_k 的准则，它通常定义为实数，如成本、时间或距离。问题(8.30)可以改写为动态递归表达式(dynamic recurrence expression)[528]。如果 $f_k(x_k,s_k)$ 代表从阶段 k 到最终阶段(n)的最优工序加工计划，则问题(8.30)的动态递归可表达如下：

$$f_k(x_k,s_k)=\min_{x_k\in D_k(s_k)}\{v_k(s_k,x_k)+f_{k+1}(x_{k+1},s_{k+1})\},\quad k=1,2,\cdots,n-1$$

尽管利用最短路径方法和动态规划可以解决这类问题，但随着问题规模的增大，许多阶段和状态都必须考虑，这将大大影响最短路径法或动态规划求解最优解时的效率。这就是众所周知的“维数爆炸”。

如果 MPP 问题(8.30)有多个目标，则问题有如下的公式：

$$\left.\begin{aligned}
\min\quad & V_1(x_1,x_2,\cdots,x_n)=\sum_{k=1}^{n} v_k^1(s_k,x_k)\\
\min\quad & V_2(x_1,x_2,\cdots,x_n)=\sum_{k=1}^{n} v_k^2(s_k,x_k)\\
& \vdots\\
\min\quad & V_p(x_1,x_2,\cdots,x_n)=\sum_{k=1}^{n} v_k^p(s_k,x_k)\\
\text{s.t.}\quad & x_k\in D_k(s_k),\quad k=1,2,\cdots,n
\end{aligned}\right\} \tag{8.31}$$

其中 p 是目标的个数。

显然，很难将问题(8.31)转化为等价的动态递归表达式。对于小规模的问题，问题(8.31)可以通过传统的多目标决策技术求解，例如目标规划。然而，如果问题的规模增大，甚至在单目标情况下，问题也会变得难以处理。

8.6.2 遗传算法

遗传子表示 对于图 8.19 中所描述的 MPP 问题，每个阶段的可选状态可以用一系列整数表示节点或状态。如果在工序计划的某个阶段上选择某项作业的一个状态，对应

节点或状态的整数被确定，该整数不大于该阶段可能的状态数。因此，如图 8.20 所示，通过连接所有阶段的状态的确定值，MPP 问题可以以状态排列表示(state permutation representation)的形式简明地编码。这种状态排列编码与 MPP 问题是 1-1 对应的，并且，随机产生工序计划的概率为 1，解码和评价也很容易。与二进制的串编码比较，状态排列编码有两大优势：

1. 对于 n 个阶段的 MPP 问题，编码的长度仅为 $n-1$。当问题规模很大时，能够节省更多的计算内存。

2. 在杂交、变异的遗传运算中，这种编码可以无需修正产生所有可行的个体。

对于一个 n 阶段的 MPP 问题的初始群体，每个个体代表一个长度为 $n-1$ 的整数排列，这些整数由相应阶段中所有可能的状态数随机产生。

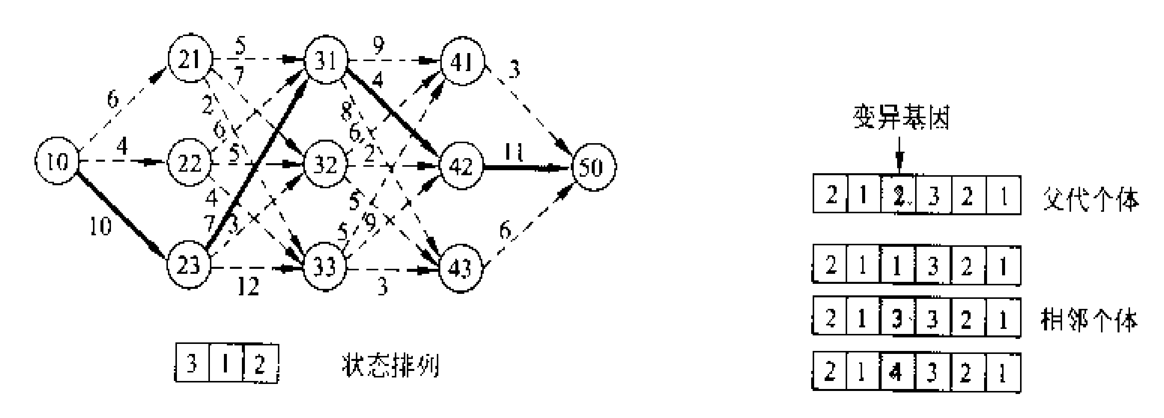


图 8.20 工序计划及其状态排列编码

图 8.21 结合邻域搜索的变异运算

遗传运算 在 Zhou 和 Gen 的方法中，仅采用变异运算，因为这很容易结合邻域搜索(neighborhood search)技术产生更适合的子代。这种混合变异(hybrid mutation)运算为进化最优解提供了很大的机会。图 8.21 举例说明了这种结合邻域搜索技术的变异运算。假设，变异的基因在阶段 3，该阶段的可能状态为 4。

评价 在多目标情况下，问题(8.31)的最优解是一个 Pareto 最优集。对于多个目标的 MPP 问题，加权和方法被用于将多目标问题转化为单目标问题，并将其倒数作为适应值：

$$\mathrm{eval}(\boldsymbol{x}) = \frac{1}{\sum_{i=1}^{p} \lambda_i V_i(\boldsymbol{x})} \tag{8.32}$$

其中，p 是目标的个数，$\lambda_i(i=1,2,\cdots,p)$ 是目标的权重系数，由 3.6 节中介绍的方法确定。

8.6.3 数例

为了验证在 MPP 问题中，遗传算法的有效性和效率。随机产生了 10 个与 Awadh，Sepehri 和 Hawaleshka[23] 给出的规模相同的 MPP 问题。所有的结果在表 8.7 中列出，

这些结果显示在求解最优解时，遗传算法比传统方法更有效。当问题规模很大时，遗传算法有更多的机会求得最优解。

表 8.7 求解 MPP 问题的不同方法比较

问题	阶段数	节点数	CPU 时间 OK/s	CPU 时间，SP/s		
				Min.	Ave.	%
1	7	24	3.01	0.03	0.04	100
2	7	27	2.84	0.03	0.04	100
3	8	38	4.31	0.06	0.09	100
4	9	37	4.25	0.06	0.12	100
5	10	47	4.48	0.09	0.27	100
6	11	53	4.58	0.34	0.45	100
7	12	63	4.69	0.32	0.48	100
8	13	72	5.08	0.61	1.03	100
9	14	79	5.19	0.49	1.01	90
10	15	89	5.35	0.97	1.28	80

注：OK 为使用 SAS/OR 软件的 out-of-kilter 算法；SP 为使用遗传算法的状态排列编码，*pop_size* 为 1000；Min. 为在所有的 20 次运算中耗时最少的 CPU 时间；Ave. 为在所有的 20 次运算中平均耗时 CPU 时间；%为在所有的 20 次运算中得到最优解的比例。

进一步，我们产生 15 个阶段和 89 个节点的 MPP 问题。对于每条边定义了两个属性：加工时间和加工成本，这些赋权值是随机产生的，分别服从 [1,50]，[1,100]上的均匀分布。两个目标是最小化总加工时间，最小化总加工成本。理想点是(91，159)，两个端点(Pareto 最优解)分别是(91,686)和(402,159)。通过遗传算法得到的结果由图 8.22 描绘。

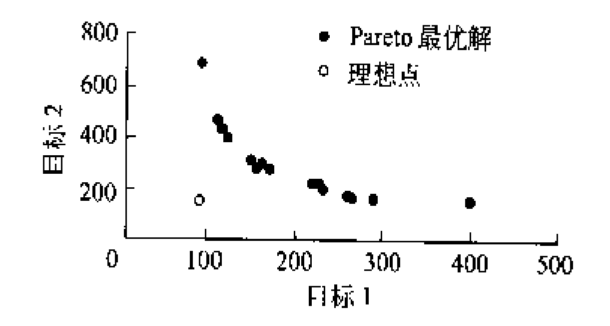


图 8.22 多目标的 MPP 问题

与枚举时 CPU 时间为 3840s，得到所有可能的真实 Pareto 最优解比较，遗传算法得到的解的 85%是真实 Pareto 最优解，且平均消耗的 CPU 时间为 2.36s。同时，图 8.22 显示，在目标空间中，所有解形成了一个 Pareto 前沿面。显然，有适应能力的超平面在引导指向理想点的搜索中扮演着重要的角色，在进化过程中迫使所有的染色体尽可能接近理想点。

8.7 网络上的 $M/G/s$ 队列设备定位

在过去的 10 年中，不确定情况下的设备定位(facility location)或随机设备定位是一个热门话题[55,75,425]。这类问题的典型要素包括顾客的定位、每个顾客的出席或缺席、需

求水平、产品价格和到达顾客的旅行时间或旅行费用。Berman 等人研究了网络上的随机队列中值模型 (stochastic queue median, SQM)[75,119]，它用于单个可移动服务点的最优定位。实际中，通常一个设施管理着多个移动服务点，而不是单个，因此，考虑多个移动服务点的情况更为合理和必要。也许有人会认为可以将较大的服务区域分为几个小的地域，这样就可以要求一台设备仅管理每个小地域中惟一的服务点。但是，问题是如何最佳地设计服务地域(见文献[56]和[57])。且必须注意到新建一个设施要比在已有设施下增加一些服务点消耗更多的资金、时间和其他资源。这些原因促使我们去研究多服务点的SQM，这可以表示为 $M/G/s$ SQM 问题。当 $s\geqslant 2$ 时，由于 $M/G/s$ 队列($M/G/s$ queue)的平均等待时间难以处理，所以这是一个难题。一个可行的方法是进行数值模拟实验，以估计系统的平均等待时间，但很难应用于大规模的网络。解决 $M/G/s$ SQM 问题的有效方法有待开发。$M/G/s$ SQM 问题的目标函数非常复杂，定位受网络的限制。$M/G/s$ SQM 问题是一个多峰值的优化问题。

在这一节中，我们将介绍 Gong, Yamazaki, Gen 和 Xu 的用于解决 $M/G/s$ SQM 问题的方法。在他们的方法中，为了估计 $M/G/s$ 队列中顾客的平均等待时间和构造 $M/G/s$ SQM 模型，采用了双时刻近似数。在这里将介绍求解 $M/G/s$ SQM 问题的全局或近似全局最优解的进化算法。运用这个模型及其求解方法，可以有效地求解大规模网络中的 $M/G/s$ SQM 问题。

8.7.1 问题的描述

我们将使用下面的网络符号：

1. 网络 $G=(V,E)$，节点集 $V=\{1,2,\cdots,n\}(|V|=n)$，边集 $E=\{(i,j)\mid i,j\in V, i\neq j\}(|E|=m)$。

2. 边 $(i,j)\in E$ 的长度给定，用 l_{ij} 表示。

3. 点 $P=(i,j:x)\in G$ 是边 (i,j) 上的点，与 i 间的距离为 x。

4. $p(P_1,P_2)$ 是 G 中两点 P_1 和 P_2 间的最短路径，$d(P_1,P_2)$ 是最短路程，即，$p(P_1,P_2)$ 的长度。

在网络 G 上，需求受网络中的节点限制。在每个节点 k，服务需求的出现服从速率为 λ_k 的 Poisson 过程，且独立于其他事件。令 λ 是系统中需求的总产生速率，h_k 为节点 $k(k=1,2,\cdots,n)$ 上需求的权重，则

$$\lambda=\sum_{k=1}^{n}\lambda_k \tag{8.33}$$

$$h_k=\frac{\lambda_k}{\lambda} \tag{8.34}$$

设施被定位在网络中的固定点 $(i,j:x)$，管理 s 个移动服务点。移动服务点被派遣

出去为需求提供服务，当所有的服务点不忙于为需求提供服务时，则停留在设施处。如果出现的需求发现所有的服务点都忙时，则插入某一队列，队列按先进先出(FIFO)的方式运作；否则，派遣闲置的服务点立即为需求提供服务。

传输以恒定的速度 ξ 发生在从设施到需求点的最短路径上。而从需求点到设施间的传输以恒定速度 ξ/β 进行，其中 β 是决定返回速度的参数。需求的服务时间是下列时间之和：

1. 从服务的设施到需求处的运行时间
2. 在需求处的台前服务时间
3. 从需求处到服务的设施的运行时间
4. 在服务出发点的台后服务时间

假设在节点 k 处的净服务时间(台前加台后)形成一更新的序列，独立于其他事件。这个净服务时间用 S_k 表示，假定具有两个有限的时刻 $\bar{S}_k$ 和 $\bar{S}_k^2$。然后这个系统可以被认为是一个遵循 FIFO 原则的 $M/G'/s$ 队列。在队列中有 n 类需求。

令 $W_q(i,j:x)$ 是 $M/G'/s$ 中第 q 个产生的需求的等待时间，$T_q(i,j:x)$ 是从设施处来的旅行时间。这样，系统对于需求的响应时间 $R_q(i,j:x)$ 可以写成：

$$R_q(i,j:x) = W_q(i,j:x) + T_q(i,j:x) \tag{8.35}$$

需求的平均响应时间定义如下：

$$\begin{aligned}\bar{R}(i,j:x) &= \lim_{N\to\infty}\frac{1}{N}\sum_{q=1}^{N}R_q(i,j:x)\\ &= \lim_{N\to\infty}\frac{1}{N}\sum_{q=1}^{N}W_q(i,j:x) + \lim_{N\to\infty}\frac{1}{N}\sum_{q=1}^{N}T_q(i,j:x)\end{aligned} \tag{8.36}$$

注意到第 q 个产生的需求属于从 1 到 n 的一类，我们有

$$\lim_{N\to\infty}\frac{1}{N}\sum_{q=1}^{N}T_q(i,j:x) = \sum_{k=1}^{n}h_k\frac{d(k,(i,j:x))}{\xi} \tag{8.37}$$

因此，等式(8.37)可以重写为

$$\bar{R}(i,j:x) = \bar{W}(M/G'(i,j:x)/s) + \sum_{k=1}^{n}h_k\frac{d(k,(i,j:x))}{\xi} \tag{8.38}$$

然后，目标是找到使 $\bar{R}(i,j:x)$ 最小的最佳位置 $(i,j:x)^*$ 或近似最优位置。

$\bar{W}(M/G'(i,j:x)/s)$ 的近似值 为了得到最佳位置 $(i,j:x)^*$，我们需要计算 $\bar{W}(M/G'(i,j:x)/s)$。注意，在等式(8.38)中，因为 FIFO 原则，需求的类别被忽略。这意味着通过使用通常的 $M/G(i,j:x)/s$ 队列，可以计算 $\bar{W}(M/G'(i,j:x)/s)$，其中 $G(i,j:x)$ 代表当设施定位在 $(i,j:k)$ 处时，所有需求服务时间的分布。分布函数 $G(i,j:k)(t)$ 如下：

$$G(i,j:k)(t) = \sum_{k=1}^{n}h_kG_k(i,j:x)(t) \tag{8.39}$$

其中，$G_k(i,j:x)(t)$ 是随机变量 $S_k+(\beta+1)[d(k,(i,j:x))/\xi]$ 的分布函数。如果不至

于混淆，$G(i,j:x)$的$(i,j:x)$将省略。G的两个开始时刻用$\bar{G}$和$\bar{G}^2$表示，给出如下：

$$\bar{G}=\sum_{k=1}^{n}h_k(\beta+1)\frac{d(k,(i,j:x))}{\xi}+\bar{S}_k \tag{8.40}$$

$$\bar{G}^2=\sum_{k=1}^{n}h_k\left((\beta+1)\frac{d(k,(i,j:x))}{\xi}\right)^2+2(\beta+1)\frac{d(k,(i,j:x))}{\xi}\bar{S}_k+\bar{S}_k^2 \tag{8.41}$$

众所周知，对于$s=1$，$\bar{W}(M/G/1)$是

$$\bar{W}(M/G/1)=\begin{cases}\dfrac{\lambda\bar{G}^2}{2(1-\rho)}, & \text{如果 } \rho<1\\ +\infty, & \text{其他}\end{cases} \tag{8.42}$$

其中，$\rho=\lambda\bar{G}$。

不幸的是，当$s\geqslant 2$时，$\bar{W}(M/G/s)$就没有如此清晰的表达式了。有人提出了近似的$\bar{W}(M/G/s)$（见文献[73]和[364]）。以下列举3个近似公式：

1. Pollaczek-Khintchine [393]

$$\bar{W}_a(M/G/s)^{(P\&K)}=\frac{1+c_G^2}{2}\bar{W}(M/M/s) \tag{8.43}$$

2. Bjorklund 和 Elldin [64]

$$\bar{W}_a(M/G/s)^{(B\&E)}=c_G^2\bar{W}(M/M/s)+(1-c_G^2)\bar{W}(M/D/s) \tag{8.44}$$

3. Kimura [364]

$$\bar{W}_a(M/G/s)^{(\text{Kimura})}=\frac{1+c_G^2}{2c_G^2/\bar{W}(M/M/s)+(1-c_G^2)/\bar{W}(M/D/s)} \tag{8.45}$$

其中，c_G^2是图G的变分系数，a代表近似值。

为了确定哪一个近似解是问题的最优解，Gong进行了许多数值模拟试验[255]。在他的试验中，使用了5个节点6条边的网络。这个网络是由Berman，Larson和Chiu[55]引人的（图8.23）。假设S_k服从均值为1.0的指数分布，$k=1,2,\cdots,5$，$\xi=1.0$，$\beta=1.0$，$h_1=0.1$，$h_2=0.35$，$h_3=0.1$，$h_4=0.35$，$h_5=0.1$。试验中参数的范围：s：2～4，λ：0.02～0.48。为了计算在任意的设施定位处$(i,j:x)$的平均响应时间，将每段弧等分成100个区间。下面的试验用于估计当设施定位在区间的开始点时的平均响应时间。独立运行6次，每次包含64000次需求，得到在每一点的平均响应时间的估计值和其95%的置信区间。

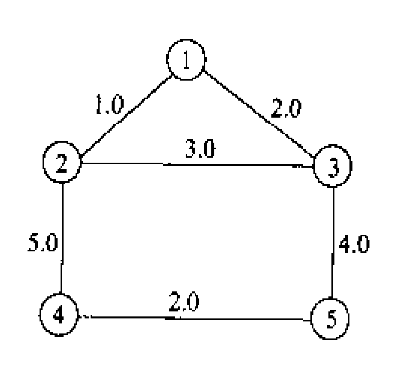


图8.23 简单的5个节点的网络问题

根据等式(8.43)～(8.45)，使用$\bar{W}(M/M/s)$的解析公式和Kimura提出的$\bar{W}(M/D/s)$的近似公式，计算三个近似平均响应时间。在$s=2$、$\lambda=0.12$的情况下的结果显示最优定位点在边(2,4)上。试验显示：(1)Pollaczek-Khintchine公式值永远小于模拟值；(2)Bjorklund-Elldin公式值更接近模拟

值；(3)Kimura 公式值几乎落在模拟值 95% 的置信区间内。其他参数的试验和其他的试验一样，产生相似的结果。这说明 $\overline{W}(M/G/s)^{(\text{Kimura})}$ 对于这类问题是适用的。表 8.8 显示了通过模拟试验，分别使用 $\overline{W}(M/G/s)^{(\text{Kimura})}$ 得到的最佳位置。在 5 个节点 7 条边、6 个节点 8 条边的网络上都进行了试验，试验结果类似表 8.8 中的结果。

表 8.8 Kimura 的近似最佳定位和最佳定位的范围

s	λ	试验后最佳定位的范围 $(i,j:x)^a$	Kimura 的近似最佳定位的范围 $(i,j:x)^b$
	0.02	(2,4 : 0.000)	(2,4 : 0.000)
2	0.12	(2,4 : 0.65～0.75)	(2,4 : 0.719)
	0.24	(2,4 : 0.70～0.75)	(2,4 : 0.725)
	0.04	(2,4 : 0.000)	(2,4 : 0.000)
4	0.24	(2,4 : 0.000)	(2,4 : 0.000)
	0.48	(2,4 : 0.25～0.50)	(2,4 : 0.298)

使用 $\overline{W}_a(M/G(i,j:x)/s)^{(\text{Kimura})}$，该问题可以描述为

$$\min \quad \overline{W}_a(M/G(i,j:x)/s)^{(\text{Kimura})} + \sum_{k=1}^{n} h_k \frac{d(k,(i,j:x))}{\xi} \tag{8.46}$$

$$\text{s.t.} \quad (i,j:x) \in G \tag{8.47}$$

8.7.2 进化计算方法

进化计算方法是一种基于群体的随机搜索方法。它使用自然基因变异和自然选择等运算进化一群所给问题的候选解。它几乎不使用有关目标函数的信息(仅仅是目标函数值)。如果有充分的计算时间(如进化世代数足够多)，它将收敛于全局最优解。实际上，有效的进化计算方法应该能在令人可接受的时间内得到近似的全局最优解。结合网络结构和进化计算概念，Gong，Yamazaki，Gen 和 Xu 提出进化计算的方法(ECA)，用于寻找 $M/G/s$ SQM 模型的近似全局最优解[247]。

染色体的表示 因为设施定位在网络上的一点，故采用网络上的点的表示：

$$F = (i,j:x) \tag{8.48}$$

这个表达式非常自然、简洁，但它需要特定的进化运算的设计。

适应值函数 对于给定的解 $F=(i,j:x)$，在 $M/G/s$ SQM 模型中，它对应的目标函数值被作为适合值。

$$\text{eval}(i,j:x) = \overline{W}(M/G(i,j:x)/s)^{(\text{Kimura})} + \sum_{k=1}^{n} h_k \frac{d(k,(i,j:x))}{\xi} \tag{8.49}$$

运行参数 下面的参数定义了在寻找近似全局最优解的过程中，ECA 需要的环境：

pop_size = 种群的大小

max_gen = 进化的最大世代数

p_c = 杂交概率

p_m = 变异概率

初始化　ECA要求一组初始的候选解以开始搜索，初始解按如下产生：

第1步：随机选择一条边$(i,j)\in E$。

第2步：在$[0,l_{ij}]$上产生一个随机实数x。

第3步：令$(i,j:x)$为一个初始解。

第4步：重复第1～3步，共pop_size次。

选择　即从当前世代中选择候选个体用于产生下一代。我们将结合最优性策略采用著名的轮盘赌策略，这对于加速收敛速度是有效的[219,245,455]。

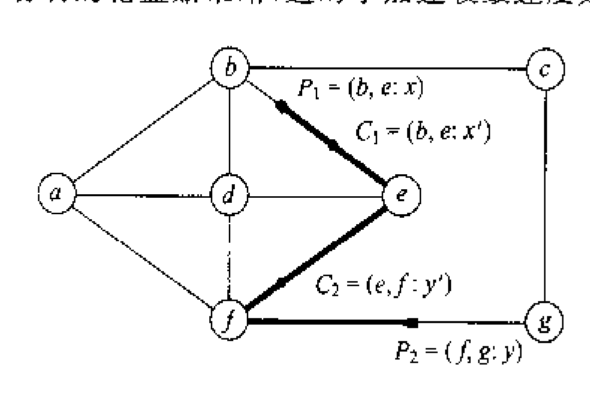


图8.24　杂交运算

遗传运算

杂交运算　令$P_1=(i_1,j_1:x_1)$和$P_2=(i_2,j_2:x_2)$是两个父代，如图8.24所示，他们杂交产生子代，$C_1=(\bar{i}_1,\bar{j}_1:\bar{x}_1)$和$C_2=(\bar{i}_2,\bar{j}_2:\bar{x}_2)$是最短路径$p(P_1,P_2)$上的两点，对应于它们的随机凸组合。

$$d(P_1,C_1)=rd(P_1,P_2)\qquad(8.50)$$

$$d(P_1,C_2)=(1-r)d(P_1,P_2)\qquad(8.51)$$

其中，r是一个随机数，$r\in(0,1)$。可以看出杂交运算永远能保证子代的可行性。

变异运算　令$P=(i,j:x)$是父代，我们使用三类变异：

(M_1) ε邻域变异：在父代的邻域中，随机产生子代$C_1=(i_1,j_1:x_1)$，即$i_1=i,j_1=j$，$x_1\in(\max\{0,x-\delta\},\min\{x+\delta,l_{ij}\})$是一个随机数，其中，$\delta$是一个小的正数。

(M_2)局部变异：在同一条边(i,j)上随机产生子代$C_2=(i_2,j_2:x_2)$，即$i_2=i,j_2=j$，$x_2\in(0,l_{ij})$是随机数。

(M_3)边变异：在另一条边上$(i,k)\in E$上随机产生子代$C_3=(i_3,j_3:x_3)$，其中k是1到n间的一个随机整数，即$i_3=i,j_3=k$，$x_3\in(0,l_{ik})$是随机数。

图8.25解释了变异运算M_1,M_2和M_3。所有的变异运算都保证了子代的可行性。

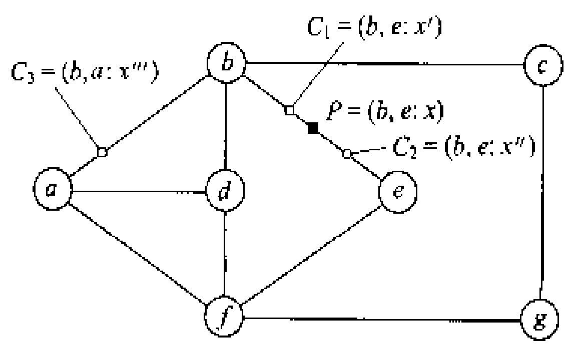


图8.25　变异运算

完整的算法　$M/G/s$ SQM的ECA算法总结如下：

第0步：初始化。

(0.1) 计算网络中所有对点间的最短

距离。

(0.2) 设置环境参数：$gen \leftarrow 0$。

(0.3) 产生初始种群。

(0.4) 计算初始种群中个体的适应值。

第 1 步：如果 $gen > max_gen$，停止；否则，继续。

第 2 步：令 $gen \leftarrow gen+1$。

第 3 步：进行繁殖。

(3.1) 在当前一代进行个体适应值的轮盘赌选择。

(3.2) 产生随机数 $r \in (0,1)$。

(3.3) 根据 r 和轮盘赌选择，在当前一代中复制个体作为临时种群池中的一员。

(3.4) 重复第(3.2)和(3.3)步 pop_size 次。

第 4 步：进行杂交。

(4.1) 产生随机数 $r \in (0,1)$。

(4.2) 如果 $r < p_c$，继续；否则，执行第(4.6)步。

(4.3) 从临时种群池中取出两父代 P_1 和 P_2。

(4.4) 根据杂交运算，产生两子代 C_1 和 C_2。

(4.5) 用子代 C_1 和 C_2 代替父代 P_1 和 P_2。

(4.6) 重复第(4.1)到(4.5)步 pop_size 次。

第 5 步：进行变异。

(5.1) 产生随机数 $r \in (0,1)$。

(5.2) 如果 $r < p_m$，继续；否则，执行第(5.6)步。

(5.3) 从临时种群池中取出父代 P。

(5.4) 交替运用变异运算(如按 $M_1 \to M_2 \to M_3 \to M_3$ 的顺序)产生子代 C。

(5.5) 用子代 C 代替父代 P。

(5.6) 重复第(5.1)到(5.5)步 pop_size 次。

第 6 步：计算临时种群池中子代的适应值。

第 7 步：应用最优性选择策略，迫使当前世代中的最佳个体进入临时种群池。

第 8 步：进入下一代。

(8.1) 用临时种群代替当前世代。

(8.2) 执行第 1 步。

8.7.3 数例

在具有 5 个节点的网络 $M/G/s$ SQM 问题上，Gong，Yamazaki，Gen 和 Xu [256] 试验了他们的算法。对于给定的一组运行参数，为了得到有关其性能的目标评价，ECA 程序执

行了30次。为了判断ECA得到的结果是否是真正的近似最优解，当$s=1$时，利用Berman的伪枚举方法得到不同比率的全局最优解；当$s\geqslant 2$时，利用数值模拟试验的结果可以估计不同比率的最优解的可能范围。表8.9给出了最优解(范围)，其中$(\cdot)^O$表示是全局最优解(范围)，$(\cdot)^b$表示ECA 30次运算的最佳解。时间是以秒为单位的实际计算时间。

表8.9　ECA得到的最优解与最佳解的比较

s	λ	最优		最佳		
		$(i,j:x)^O$	$R(\cdot)^O$	$(i,j:x)^b$	$R(\cdot)^b$	时间/s
1	0.01	(2,4:0.000)	3.230	(2,4:0.000)	3.236	1.01
	0.06	(2,4:1.614)	5.893	(2,4:1.612)	5.946	1.02
	0.12	(2,4:0.934)	23.525	(2,4:0.913)	23.871	1.01
2	0.02	(2,4:0.000)	2.877	(2,4:0.000)	2.877	1.42
	0.12	(2,4:0.65～0.75)	3.88～3.92	(2,4:0.719)	3.895	1.42
	0.24	(2,4:0.70～0.75)	12.0～12.6	(2,4:0.725)	12.466	1.42
4	0.04	(2,4:0.000)	2.850	(2,4:0.000)	2.850	1.83
	0.24	(2,4:0.000)	3.071	(2,4:0.000)	3.071	1.84
	0.48	(2,4:0.25～0.50)	6.9～7.2	(2,4:0.298)	7.027	1.82

ECA的运行参数设置如下：$pop_size=20$，$\max_gen=600$，$p_c=0.4$，$p_m=0.2$。表8.9给出了运行的结果，表8.10给出了30次运算的平均性能，其中$(\cdot)^a$表示是30次运算的平均值。从表8.9中可以看出：对于所有的λ和s，ECA总能找到全局或近似的全局最优解。这些结果表明ECA的全局最优化能力。从表8.10中可以看到，对于所有给定的λ和s，ECA的30次运算均收敛于全局或近似的全局最优解(30次运算的适应值均为0)。这说明了ECA具有良好的收敛性。

表8.10　在5个节点的网络上30次运行的平均性能

s	λ	$(i,j:x)^a$	$R(\cdot)^a$	方差	计算时间/s
1	0.01	(2,4:0.000)	3.236	0.000	1.01
	0.06	(2,4:1.612)	5.946	0.000	1.02
	0.12	(2,4:0.913)	23.871	0.000	1.01
2	0.02	(2,4:0.000)	2.877	0.000	1.42
	0.12	(2,4:0.719)	3.895	0.000	1.42
	0.24	(2,4:0.725)	12.466	0.000	1.42
4	0.04	(2,4:0.000)	2.850	0.000	1.83
	0.24	(2,4:0.000)	3.071	0.000	1.84
	0.48	(2,4:0.298)	7.027	0.000	1.82

为了在大型网络问题上试验 $M/G/s$ SQM 模型和 ECA，如下产生一个具有 64 个节点 188 条边的网络。

1. 产生 64 个节点的完全图，如果 $i=j$，则边 (i,j) 的长度是一个很大的常数 10000，否则是(10,80)上的一个随机整数。

2. 随机选择一段弧，如果不改变图的连通性，则删除它。

3. 重复步骤 2，直至只剩下 188 条边。

假设网络中每个节点的权重是相等的$\left(\text{例如 } h_k=\frac{1}{64}\right)$，需求服务时间的第一、第二时刻为 $\bar{S}_k=\bar{S}_k^2=1.0(k=1,2,\cdots,64)$，服务的旅行速度为 $\xi=10.0,\beta=1$。

ECA 的运行参数设置如下：$pop_size=40$，$\max_gen=600$，$p_c=0.4$，$p_m=0.2$。表 8.11比较了在 $s=1$ 的情况下，ECA 与 Berman 寻求最优解的计算时间。在 $\lambda=0.12$，$s=2$ 的情况下，64 个节点网络上的 $M/G/s$ SQM 模型可以使用 ECA 求解。计算结果：最优解是(43,35 : 0.0)，最佳适应值(最短平均响应时间)18.041，计算时间是 36.0s。为了检查这是否是近似最优解。我们进行了数值模拟试验。对于边(43,35)上给定的点模拟试验需要 39.4s，所以，如果每条边分成相等的 100 个区间，模拟试验在整个网络每个区间上进行，那么它将耗时 6.56×10^5s，即一个多星期。由于需要花费大量的计算时间，故试验仅在边(43,35)上进行。

表 8.11 ECA 与伪枚举法的比较($s=1$)

λ	最优			最佳		
	$(i,j:x)$	$\bar{R}(\cdot)$	时间/s	$(i,j:x)$	$\bar{R}(\cdot)$	时间/s
0.01	(45,9 : 0.00)	7.06	42.64	(45,35 : 0.00)	7.06	32.25
0.02	(23,43 : 8.37)	9.00	42.64	(23,43 : 8.37)	9.00	32.25
0.03	(23,43 : 9.56)	11.06	42.64	(23,43 : 9.54)	11.66	32.25
0.05	(23,43 : 8.76)	22.05	42.64	(23,43 : 8.76)	22.05	32.25
0.07	(45,9 : 0.00)	65.87	42.64	(45,9 : 0.00)	65.87	32.25

表 8.10 表明，在 64 个节点的网络上 ECA 比伪枚举方法花费更少的计算时间找到近似最优位置。有一点值得注意，ECA 的计算时间比达到近似最优位置的实际计算时间多，因为，为了确保 ECA 的收敛性，进化 600 世代有一点长了。从结果看，近似的平均响应时间与模拟结果是一致的；尽管 ECA 得到的最佳位置只花费了 35.52s，但与模拟的结果是一样的。因为 64 个节点的网络是随机产生的，因此提出的 $M/G/s$ SQM 模型和 ECA 过程对于一般的大型网络问题是适用的。更具体的试验详见文献[255]。

第9章 制造元设计

9.1 引言

Mitrofanov 于 1959 年创立的成组技术现在引起了理论工作者和实际工作者相当大的兴趣。成组技术是辨别和发掘一组目标特征之间相似性的一种方法。单元制造(cellular manufacturing)是运用成组技术组织机器单元来加工一组零部件[81]。成组技术应用的重要领域是分类和编码、加工计划、零部件族群和机器制造元设计(manufacturing cell design)、成组技术布局和成组计划。在近来的自动化制造环境里，单元结构是研究的重点。柔性制造系统（flexible manufacturing system），计算机集成制造（computer-integrated manufacturing)，准时生产（just-in-time)系统是自动化制造单元的常见例子，Wemmerlov 和 Johnson[658]总结了公司建立单元制造的理由。

- 减少总生产时间
- 减少在加工作业（work-in-process)产品的库存
- 提高零部件或产品的质量
- 减少顾客订单的响应时间
- 减少移动时间
- 增加制造柔性

增加工作柔性与合作，减少挫折，改进工作地位和工作安全往往是雇员或工人的利益所在[185]。成组技术中将机器和零部件组成制造单元的过程称为制造元设计(MCD)(manufacturing cell design)，已经证明这是一个 NP-完全问题。在过去的二十多年里，产生了许多方法来解决这个问题。最近，应用进化搜索算法对制造元设计问题也作了许多尝试，模拟煺火算法[69,116,285,421,639]，禁忌搜索[598]和遗传算法[62,274,333,334,350,362,463,639]是非常有效的进化探索方法，其中，遗传算法为解决这类复杂问题提供了最好的柔性和广度。在这一章中，我们回顾应用遗传算法解决制造元设计的最新文献并详细介绍遗传算法解决这类问题的过程。在 9.2 节描述制造元设计问题的基本概念，9.3 节回顾制造元设计的传统方法，9.4 节介绍用遗传算法求解制造元设计问题的几种方法。最后，在 9.5 节我们讨论如何运用遗传算法求解可选加工计划的制造元设计问题。

9.2 制造元设计

成组技术不同于传统的车间作业安排。如果生产的品种数量小而需求量大(即大批量),按产品布局是对大批量生产最有效的方法。在流水线上,产品布局由一组不同的机器组成,每一条流水线用于生产一种指定的产品,如表 9.1(a)所示。由于一条流水线用于生产一种产品以及相关的机器安置在流水线的适当位置,因此,装置更换、在加工作业、生产时间、物料传送都大大减少。一旦零部件品种的数量增加,而零部件的需求量减少(即中小等规模),由于必须使用重置的机器和额外的空间,把特定的生产线指定生产每一产品将不再是经济的了。因此,常常使用功能性的或*车间作业布局* (job-shop layout),在这里相似的机器和相应的熟练员工被一起置于功能区域,如图 9.1(b)所示。通过所有产品而不是单个产品共享机器,使得机器利用率提高,从而使同类机器数量减少。

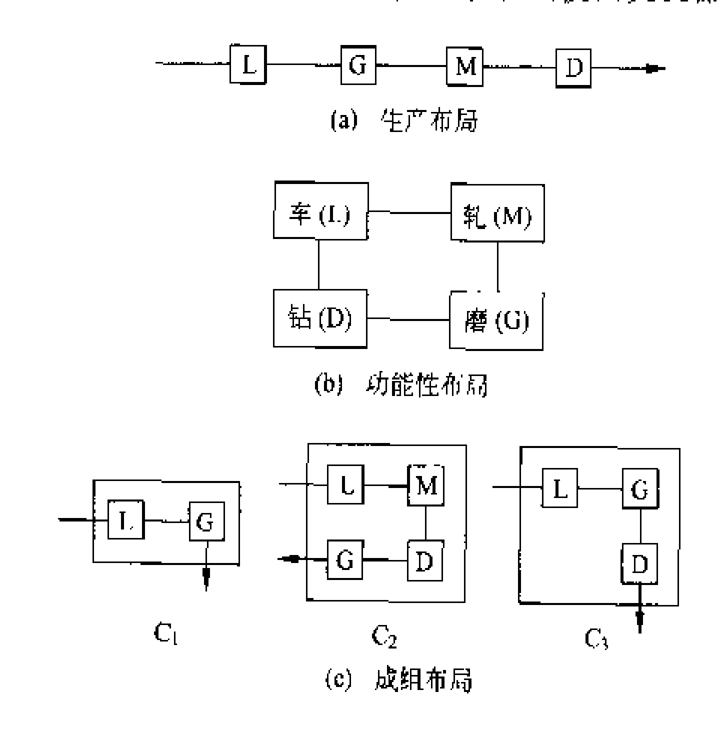


图 9.1 物料流程图

功能性布局(functional layout)为多品种生产(即许多不同路径的生产线)提供了最大的柔性。由于一个零部件的加工能够容易地访问每一个必要的加工功能组,新零部件的加工可以毫无问题地引入,而不像产品布局,机器必须重新排列或布局。典型的作业车间具有如下缺点:由于生产品种的多样性而增加装置更换时间;由于加工零部件在整个作业场地的传送而增加物料传送量;由于品种的数量及其资源共享造成作业计划难度的增加;由于不同零部件要求相同的机器造成加工过程复杂,并引起加工过程延迟。

为克服功能布局的缺点而同时保持生产流程的柔性,以便从小批量到中等规模生产

多品种产品，可以利用成组技术原理进行布局。成组布局可以看作是车间作业布局和产品布局的结合，即一组不同类型的机器组合在一起，对一组相似的零部件进行加工处理，如图 9.1(c)，此处，机器往往是按直线型或 U 字型排列。因此，一族群零部件的加工只需要一个非常小的车间作业布局，其中只存在一个主要的流程模式。

在成组布局和单元制造系统的发展和实施中，制造元设计问题是一项非常重要的工作。一族群零部件的加工可以由若干组零部件组成，这些零部件的处理过程、原材料和工具可以是相似的，有时可能是相同的。此时，将生产一族群零部件的机器聚集起来就形成一个制造元。为了说明制造元设计问题，可以考虑图 9.2(a)所示的机器和零部件矩阵。如果第 i 种零部件由第 k 种机器加工，则矩阵的元素(i,k)为 1；否则为 0。图 9.2(a)行和列各元素重新排列之后最后可得到图 9.2(b)的结果。这些最终形成的组群对应于制造系统的单元，即三个机器单元，$C_1=\{2,5\}$，$C_2=\{3,4,6\}$，$C_3=\{1,7\}$，以及三个零部件族群$F_1=\{1,7\}$，$F_2=\{3,4,6\}$，$F_3=\{2,5\}$。

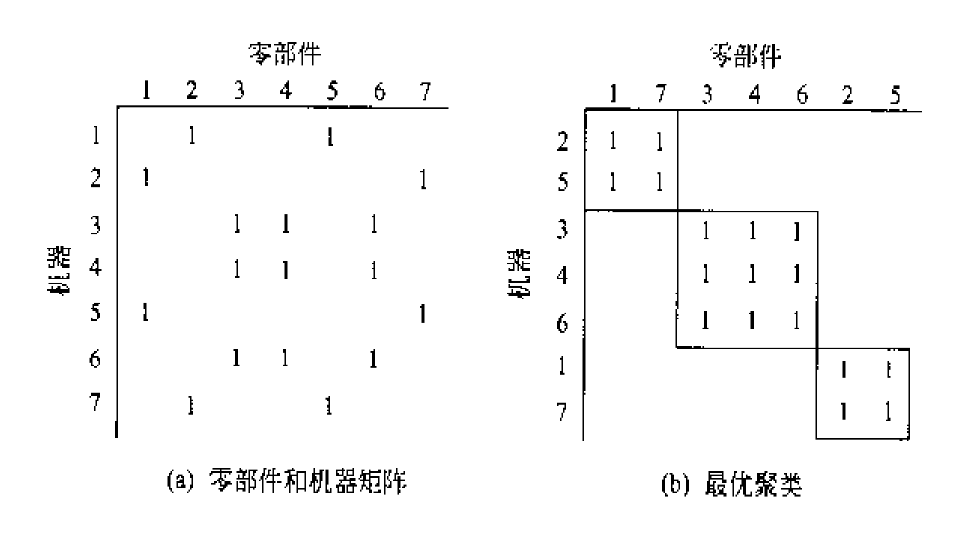


图 9.2　零部件和机器矩阵及其最优聚类

9.3　传统的制造元设计方法

制造元设计可以归为组合优化问题，此处，零部件和机器关联矩阵含有 n 行 m 列，分别包含 $n!$ 和 $m!$ 种不同的组合。因此，大多数制造元设计的方法是基于相似系数矩阵、数组、数学规划、图和网络以及其他的启发式方法。这些方法的目标多数是单元间的移动极小化或者是考虑车间场地因素的前提下单元的独立性极大化，这些因素包括生产量、单元规模，可选的加工计划或加工时间等。

然而，文献中的大多数算法仅仅利用了预先指定的单一和固定作业路径的零部件和机器关联矩阵。相对来说，很少有研究者讨论如何利用可选加工作业的问题，以及完备的固定可选作业路径问题，或在确定单元和族群时的机器冗余问题。固定的作业路径常常

用于功能布局中机器利用的最大化而不只是单元制造的利用。包括可选作业和多作业路径，这些方法多数是使用数学规划，但在求解大规模问题时候往往受到限制。

同样，这些方法的大多数都假定每一零部件的需求量都是相等的。然而，在大多数情况下，零部件的需求量是不相等时，此时，这种假设下形成的机器单元和零部件族群将会变得效率非常低下。尽管有一些方法提供了更好的结果，但是，还没有一种技术表明可以为大多数应用问题提供最好的解决方案。这些方法的典型是包含单一准则的目标标准，不允许评价函数的交换和约束条件的选择使用。并且，多数的制造元设计方法不能识别所有的自然聚类，也不能找到带有聚类约束的解决方案。大多数方法不能包含其他的制造信息，例如，零部件的需求、可选作业和机器冗余等。

9.3.1 相似系数方法

相似系数方法 (similarity coefficient methods)利用每对物体的相似系数和某个指定的阈值把他们组合成一些单元。该阈值是两个或两个以上的串结合在一起的相似水平。例如：机器 i 和 j 间的相似系数 S_{ij} 为：

$$S_{ij} = \frac{N_{ij}}{N_i + N_j - N_{ij}} \tag{9.1}$$

其中 N_i 是访问机器 i 的零部件数量，N_{ij} 是共同访问机器 i 和 j 零部件的数量。为了说明式(9.1)的相似系数，结合图 9.2(a)的系数矩阵，每一对机器间的相似系数计算如下：

$$S_{13} = \frac{N_{13}}{N_1 + N_3 - N_{13}} = \frac{0}{2+3-0} = 0$$

$$S_{46} = \frac{N_{46}}{N_4 + N_6 - N_{46}} = \frac{3}{3+3-3} = 1$$

当相似系数的阈值为 $S_{ij}=1$ 时，我们得到两个单元和相应的零部件族群如下：

$$S_{25} = 1,\quad S_{34} = S_{36} = S_{46} = 4,\quad S_{17} = 1$$

$$S_{12} = S_{15} = S_{23} = S_{24} = S_{37} = 0$$

McAuley[446]首先使用单一链聚类分析(SLCA)方法，这种分析方法基于相似系数组合零部件和机器。此种系数定义为“访问两台机器中任一台机器的零部件数量除以共同访问两台机器的零部件数量”。此后，Seifoddini 和 Wolfe[569]，Gupta 和 Seifoddini[273]提出了修改和改进的相似系数计算方法。这些方法为用户提供分层次的解决方案，以便从中选择最好的解决方案。

9.3.2 基于数组的方法

基于数组的方法 (array-based methods)将零部件和机器关联矩阵作为输入，重复地重新排列矩阵的行和列，以形成最终的组合。McCormick 等人[447]提出一个基于二次指

派问题的成组方法，称之为联接能量算法(bond energy algorithm)(EBA)。联接能量定义为关联矩阵中任何两个邻近元素间的键力总和，目标是使得矩阵的联接能量最大化，如果存在的话，就产生一个块对角矩阵。Bhat 和 Haupt[61]扩展了联接能量模型来提高该方法的计算效率。King[366,367]提出了一种基于数组的秩排序聚类法(ROC)，这种方法可以描述如下：

第1步：对关联矩阵的每行每列赋予二元权重

第2步：将关联矩阵每行每列的二元权重转化为等价的十进制值

第3步：交替地重新排列行和列以减少数量级，直到形成最后的机器和零部件的分组。

King 和 Nakornchai[368]提出了一种扩展的秩排序聚类法用于解决大规模的问题。稍后 Chan 和 Milner[91]，Chandrasekharan 和 Rajagopalan[93]，Kusiak 和 Chow[385]也对秩排序聚类法进行了一些改进。

9.3.3 数学规划方法

数学规划模型是制造元设计最著名方法之一。为解决 p-中值模型(p-median model)，Jensen[336]提出了一种动态规划模型。在每一对象仅仅一次性地划分给某个组群的条件下，并假定组群的个数(p 个)为已知，模型的分组度量是求被划分到同一组中对象的相似值总和为最大。Kusiak 和 Heragu[388]为制造元设计提出了一种应用整数规划的 p-中值模型，阐明了考虑可选作业计划的重要性。然而，p-中值模型对单元的规模进行控制。基于此，Srinivasan，Narendran 和 Mahadevan[582]提出了一种改进的 p-中值模型。Ham和 Han[278]提出了一种在聚类和编码中形成零部件族群的整数目标规划模型，采用 Minkowski 距离测度，该模型的目标是按字典序最小化这些距离。为了考虑可选作业计划的制造元设计，Kasilingam 和 Lashkari[348]提出了一种非线性 0-1 型整数规划模型。基于加工需求和处理时间，该模型的目标是使得定义在零部件和机器间的兼容性指标最大化。最近，Adil，Ragamani，和 Strong[6]为考虑可选加工作业计划的制造元设计，提出了一种非线性整数规划模型，该模型综合考虑了极小化加权的单元间移动次数和无效移动次数的总和及其评价程序。所谓无效移动是指划分到某个单元的机器对这个单元中零部件的加工是不需要的。由于计算的复杂性，对于大多数上述方法在解决大规模问题时是有其局限性的。

9.3.4 图与网络方法

Rajagopalan 和 Batra 提出了一种利用图分割(graph partitioning)的制造元设计方法。在他们的算法中，顶点表示机器，每一条边表示由两台机器加工的零部件，两台机器表示边的末端。Lee 和 Garcia-Diaz[397]把制造元设计转化为网络流问题，使用 Bertsekas 和 Tseng[58]提出的对偶算法来解决制造元设计，这种算法比 Kusiak 和 Heragu[388]的

p-中值模型计算更有效。其他的图和网络方法包括：Ng[483]的最小生成树算法、Askin 和 Chiu[21]的启发式图分割方法、Vohra 等人[642]的 Gomory-Hu 改进算法的网络方法。最近，在制造元设计问题上，为最小化单元间零部件的移动，Lee 和 Garcia-Diaz[398]提出了一种基于网络流的三阶段方法。

9.4 遗传算法方法

9.4.1 遗传子表示和遗传算子

为了使用遗传算法解决制造元设计问题，遗传子表示的方案决定了在遗传算法中问题如何构造以及遗传操作如何运作。因此，选择合适的遗传子表示是用遗传算法解决制造元设计问题的第一步。对于制造元设计有二种遗传子表示方案：(1)基于单元数的遗传子表示(cell number-based representation)；(2)基于次序的遗传子表示 (order-based representation)。

基于单元数的遗传子表示：制造元设计问题常见的整数规划描述由以下变量和约束组成，这些变量和约束确保每一机器和零部件分别仅仅指派给惟一的单元和族群，即：

$$x_{ic} = \begin{cases} 1, & \text{如果机器 } i \text{ 指派到单元 } c \\ 0, & \text{否则} \end{cases}$$

$$y_{jc} = \begin{cases} 1, & \text{如果零部件 } j \text{ 被指派到零部件家族 } c \\ 0, & \text{否则} \end{cases}$$

$$\sum_{c=1}^{k} x_{ic} = 1, \qquad i = 1,2,\cdots,m$$

$$\sum_{c=1}^{k} y_{jc} = 1, \qquad j = 1,2,\cdots,m$$

其中：k——给定单元(族群)个数；

m——机器台数；

n——零部件个数。

对于 k 个单元的 m 台机器/n 个零部件的问题，总数有 $k(m+n)$个变量和 $m+n$ 个约束。显然，当机器和零部件的数量增大时，模型将变得太大而内存不足或者难以计算处理[397]。为了克服这些局限性，Joines 等人[333,336]，Venugopal 和 Narendaran[639]，Moon，Kim 和 Gen[463]分别各自提出了带有如下变量集的整数规划模型：

$$x_i = c, \quad \text{机器 } i \text{ 被指派给单元 } c$$

$$y_j = c, \quad \text{零部件 } j \text{ 被指派给族群 } c$$

这个模型利用一个 k 到$(m+n)$的因子减少了变量的数目，并运用集合的概念除去了约束

条件。每个零部件和机器的变量数等于指派的族群或单元的个数。在这个方案中染色体或基因值代表了单元编号，基因的位置相应于机器编号。这种染色体表示的方案如下所示：

[3 1 1 2 3 1 2]

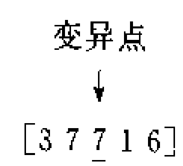
其中，染色体的长度取决于生产过程中机器的数量，每一个基因值代表机器被指派的单元号。在此，我们考虑 7 台机器和 3 个单元的问题，机器 2,3,6 被指派到单元 1；机器 4,7 被指派到单元 2；机器 1,5 被指派到单元 3。

基于次序的遗传子表示　基于次序的遗传子表示广泛地应用于遗传算法。对于带有序的问题已经发展了许多这类遗传子表示方案，其中，排列表示或许是最自然的路径表示。Cheng 等人[101]，Billo，Tate，Bidanda[63]，Kazerooni[350]，Kazerooni，Luong 和 Abhary[351]将它应用于制造元设计问题。例如，有 7 台机器(或零部件)和 3 个单元的问题，其染色体可以描述如下：

[2 7 4 * 3 1 6 * 2 5]

在这个染色体中，单元的位置由星号(*)来识别，他指明了包含在单元内机器数的中止点。第一个单元包含机器 2,7,4；第二个单元包含机器 3,1,6；第三个单元包含机器 2,5。

制造元设计问题的遗传算法中，杂交和变异是主要的遗传操作。在基于单元数的遗传子表示中，常常使用简单的杂交操作，如单点分割杂交和两分割点杂交。然而，对于基于次序的遗传子表示，已设计了若干种杂交操作，例如部分映射杂交(PMX)，次序杂交(OX)，循环杂交(CX)，边的再重组杂交(ER)，等等[219]。其中，部分映射杂交(PMX)操作被广泛应用于基于次序的遗传子表示的制造元设计问题[333]。变异按随机变动执行，染色体的每个基因以一定的变异概率随机的选定进行变异操作，这种变异操作的说明如下：



在这个染色体中第三个基因被选定变异，其基因值由[1,10]内的随机整数代替。如果随机产生的值为 4，则变异后的子代为：

[3 7 4 1 6]

在基于次序的遗传子表示中，遗传操作随机选定一个染色体中的两个基因，交换他们的位置，或者随机选定一个基因，嵌入到一个随机选定的位置。有一些变异操作已被应用于基于次序的遗传子表示，例如：倒置变异，嵌入变异，位移变异，互反变异等。

根据以往经验，Joines[333]认为以下 5 种基于次序的操作非常适合于制造元设计问题。

1. 变异操作

　(a) 单点变异

(b) 交换变异

(c) 倒置变异

2. 杂交操作

(a) 部分映射杂交(PMX)

(b) 均匀基于次序的杂交(UOX)

9.4.2 Joines 基于次序的方法

Joines[333]运用两个不同的基于次序的遗传算法确定机器和零部件的最好排列,使得总的联接能量 (bond energy)最大。给定特定的行排列(π)和列排列(ρ),总的联接能量(TBE)定义如下:

$$\text{TBE}(\pi,\rho) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}(a_{i,j-1} + a_{i,j+1} + a_{i-1,j} + a_{i+1,j}) \tag{9.2}$$

其中

$$a_{0,j} = a_{m+1,j} = a_{i,0} = a_{i,n+1} = 0$$

当一个正元素(即需要在机器 i 上的进行加工零部件 j)与同一零部件 j 上运行的其他机器(即行序列上机器 i 的任一侧)安排在一起,或当要求在同一台机器 i 上操作的其他零部件与列序列上零部件 j 的任意一侧(即左侧或右侧)安排在一起时,联接就形成了。任何给定元素的最大联接能量为 4,即 1 紧靠该元素的上下左右。由于对角线并不包括在总联接能量的计算中,即行和列之间是独立的,因此,这个问题可以分解为行联接能量(RBE)和列联接能量 (CBE),总联接能量 TBE 等于行联接能量(RBE)和列联接能量(CBE)的总和。不失一般性,可以先行排序后列排序。出于求解考虑,机器(行)先组成机器单元,然后零部件(列)反复排列。

$$\max_{\pi}\text{RBE}(\pi) = \sum_{i=1}^{M-1}\sum_{j=1}^{N} a_{\pi(i),j}\, a_{\pi(i+1),j}$$

遗传操作 在 TSP 问题上十分有效的基于次序的遗传操作对于制造元设计问题并不一定有效。基于次序的杂交(OX) 已被表明比部分映射杂交(PMX)和循环杂交(CX)好。然而,基于次序的杂交并不适合于聚类问题,因为 TSP 问题允许旅行商人可以从任一城市开始,只要访问所有每一个城市,然后回到城市的起点。所以,两个染色体[1 4 0 5 2 3 6]和[2 3 6 1 4 0 5]可以看作是相同的,基于次序的杂交可以利用这种特征探索新的解。然而,在聚类问题中,个体的最后一个元素并不能如同 TSP 问题中那样,认为它与第一个元素相邻。因此,部分映射杂交操作经常取代基于次序的杂交操作。

计算结果 根据文献报道,基于次序的遗传算法应用于两类制造元数据集。遗传算法使用基于规范化的几何分布等级方案(见 9.4.4 节),一个最优性模型,以及如前一节所述的基于次序的遗传操作。表 9.1 中列出了为两类数据集设置的遗传算法参数。

表9.1所给出遗传操作次数是确定的，例如：单点变异应用于四个随机选定的父代。基于次序遗传算法的聚类工具应用于两类检验数据集，第一类数据集由King和Nakornchai[368]给出，它是一个包含了两个瓶颈机器(即机器5和7，如图9.3所示)的16×43问题，解决方案如图9.4所示。第二类数据集是一个20×35矩阵，来源于Burbidge[80]。原始数据集和基于次序的遗传算法的解如图9.5和图9.6所示，单元边界的判断取决于视觉检查。基于次序的遗传算法可以求得总联接能量意义上的最优解。然而，制造元设计者必须从视觉上检查最终的输出结果以获得每一个单元的实际成员。除了King的数据集外，这种判断并不困难，King的数据集的难度在于难以确定瓶颈机器应被指派给哪一个单元。最终，这种算法不需要产生解的方案，这种方案就是沿着对角线形成的单元。实际上，机器是分别地从零部件到族群划分成单元的。即使这种基于次序的遗传算法对所有的数据集都运行得很好，它也不能同时确定单元和族群，并且，需要视觉来判断确定输出的单元和族群。

表9.1　在基于次序的经历中使用的参数

参　　数	操作的评价或数量
单点变异	4
交换变异	4
倒置变异	4
部分映射杂交	6
均匀基于次序的杂交	6
最优个体的选择概率，q	0.1
最大世代数，T	10000

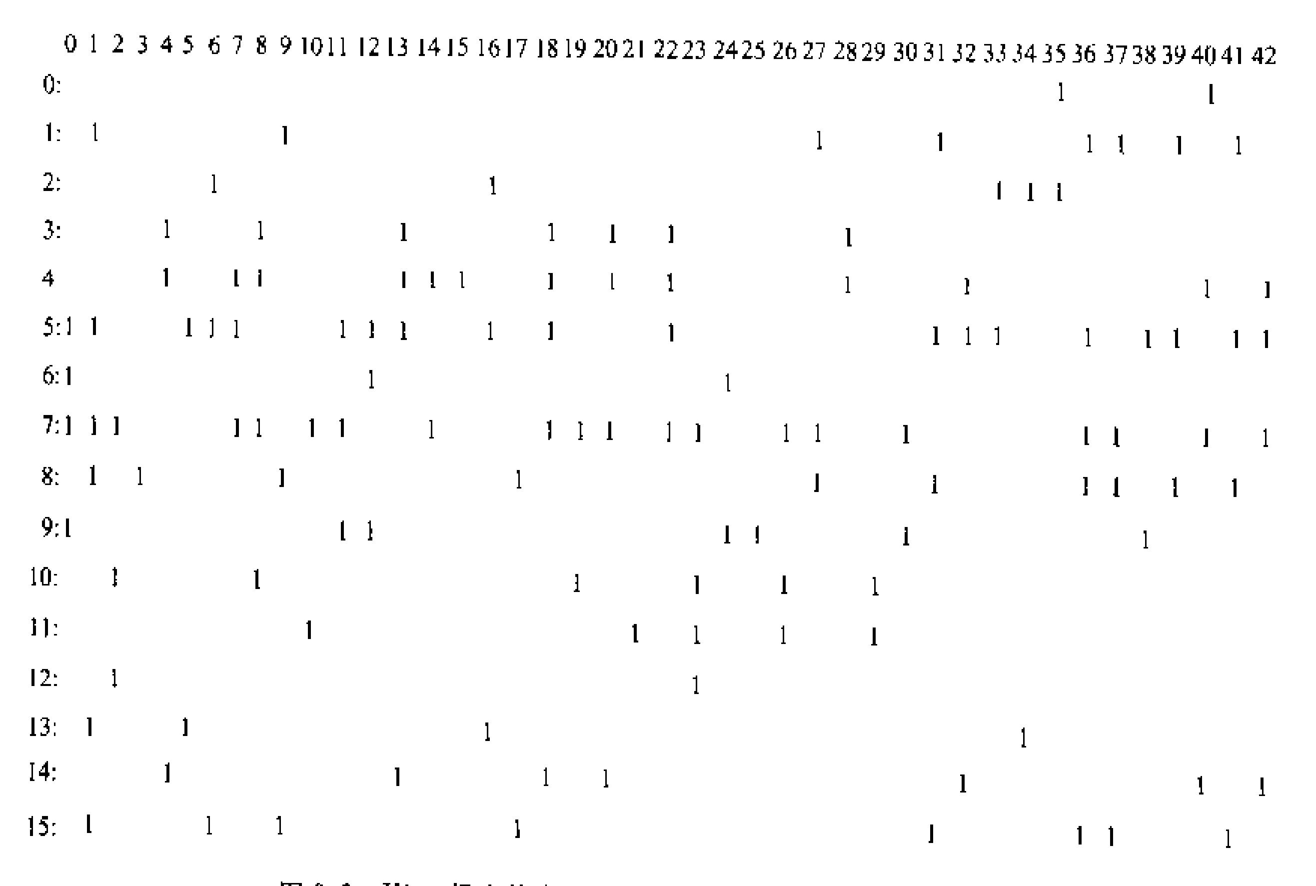


图9.3　King提出的16×43零部件与机器关联矩阵原始数据

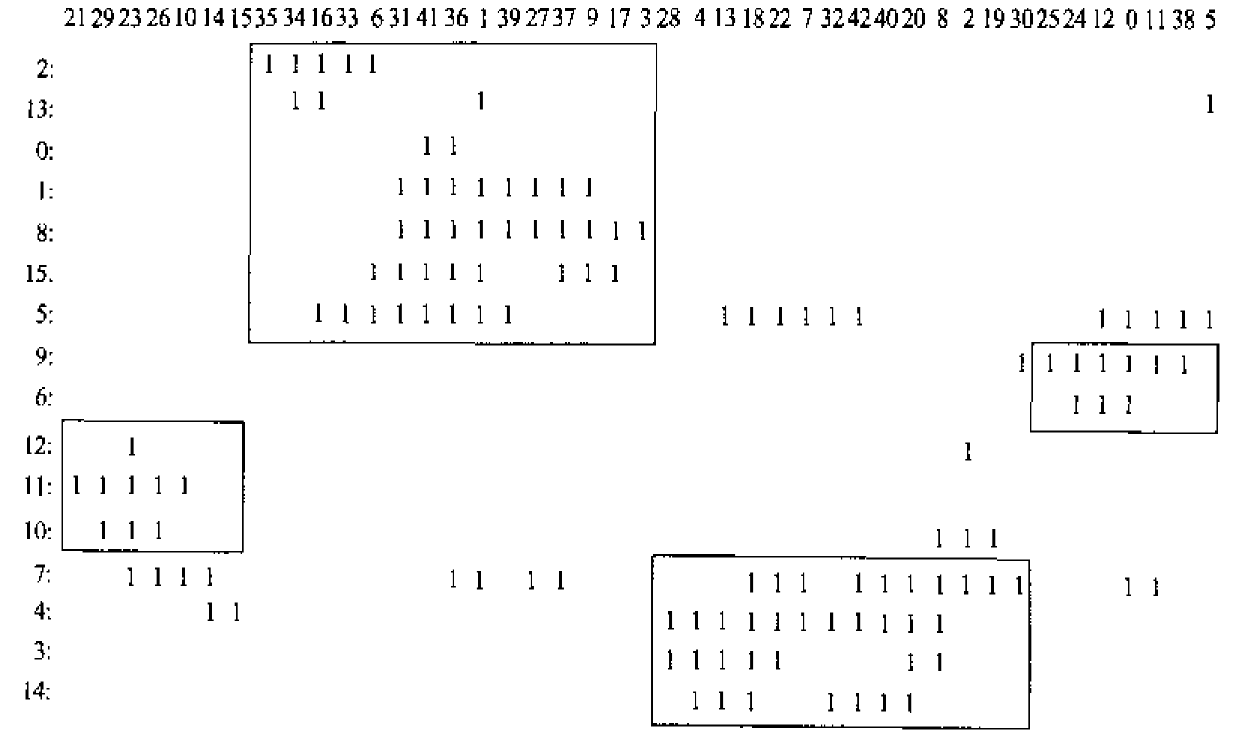


图 9.4　总联接能量为 154 的 King 数据集的基于次序的遗传算法解

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
0:	1		1																	1			1		1										
1:		1					1			1		1	1					1						1			1				1				
2:	1		1		1										1		1												1		1				
3:		1					1					1	1											1			1								
4:								1						1		1																		1	
5:								1						1		1			1			1				1								1	
6:	1		1		1										1		1			1			1												
7:	1		1		1										1		1			1			1		1				1						
8:								1						1					1			1				1									
9:								1						1		1			1			1				1									
10:				1		1			1		1										1							1		1		1			1
11:				1		1			1		1										1												1		
12:		1										1	1											1											
13:		1					1			1		1	1					1						1			1				1				
14:				1		1			1		1										1							1		1					
15:				1		1			1		1										1							1		1		1			
16:	1		1		1										1		1			1			1		1				1						
17:		1								1		1	1					1						1							1				
18:				1					1		1										1							1		1		1			
19:								1						1					1				1			1									

图 9.5　Burbidge 提出的 20×35 零部件与机器关联矩阵原始数据

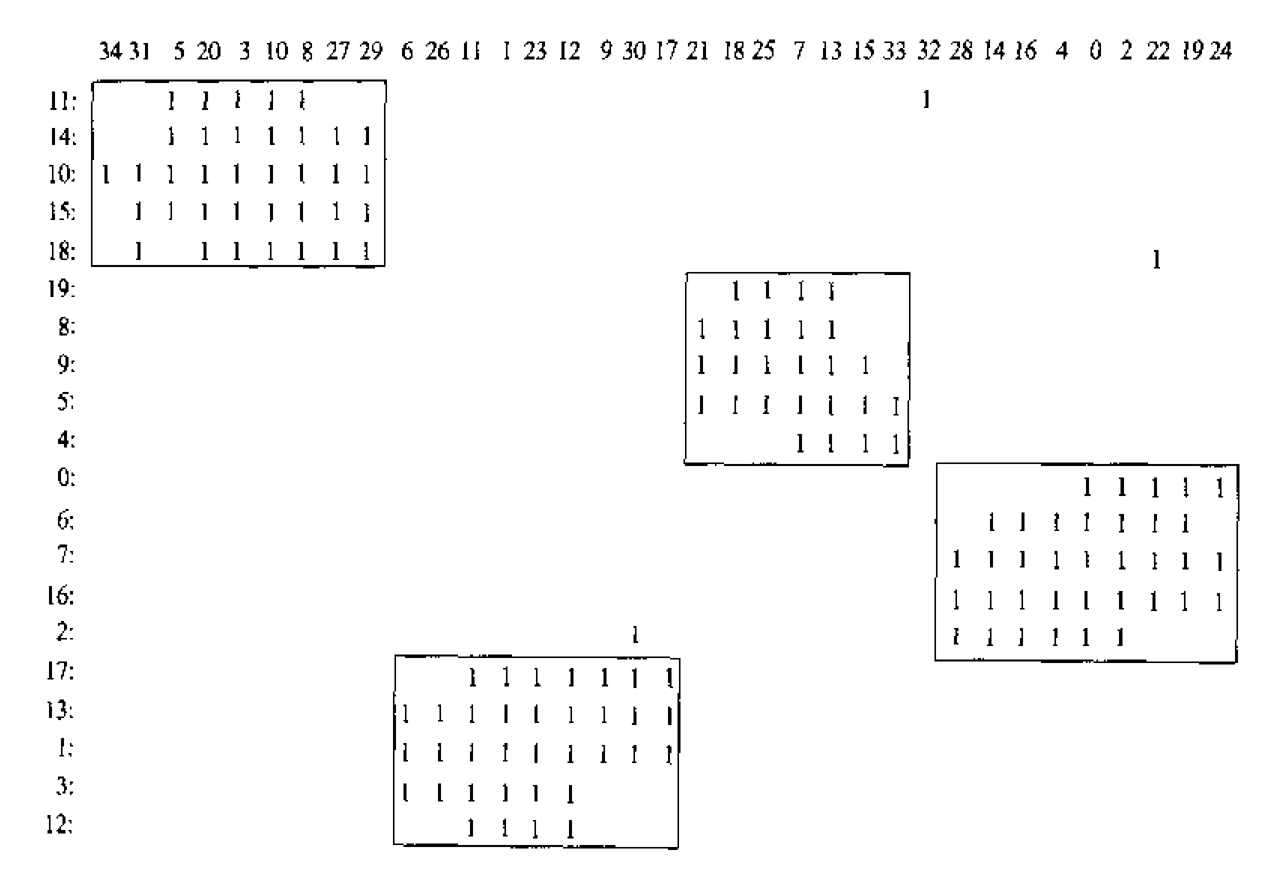


图9.6　总联接能量为211的Burbidge数据集的基于次序的遗传算法解

9.4.3　Moon和Kim的方法

聚类问题的数学规划方法是非线性或线性规划问题[69,386,388]。这些方法受限于三个关键因素。第一，由于目标函数的非线性形式，大多数的方法不能同时地将机器分组成单元和零部件分组成族群[69]。第二，机器单元的个数必须制定一个优先级，这就影响分组过程以及潜在地阻碍数据中单元的构成。第三，由于变量限制于整数值，那么，对于实际规模的问题，这些模型大多数都是计算上难以处理的[397]。然而，运用基于单元数遗传子表示的遗传算法，可以将通常整数规划形式的制造元设计问题的变量简化。Moon和Kim[463]对机器制造元设计提出用遗传算法来解决其整数规划模型，使在同一个单元内机器之间的加工零部件流最大化。因此，高度关联的一批机器应该被包括在同一个单元内。两台机器间零部件流的计算基于：加工计划，生产量，原材料处理时间和每一行程移动装置的有效容量。

问题描述　外形不同的零部件可能要求相似的制造处理。由于族群中的成员都要求相似的制造处理，可以建立一个机器单元对该族群进行制造处理。按照加工计划单，一组零部件以及其加工计划和生产量如表9.2所示。当使用表9.2的数据作为输入的时候，机器之间的关系可以用赋权弧表示两台机器间的物流。机器之间物流的计算基于：加工

计划，生产量和每一行程移动装置的有效容量。例如，生产中每一行程移动装置的有效容量为 2 单位，那么，物流图示可以描述如图 9.7 所示。图中，节点和弧分别表示机器和机器间的关系，这种关系定义为两台机器间总的零部件流量。设计机器单元本质上意味着识别和使用机器间的关系，达到单元间的移动最小化，单元内部能同时处理的零部件总量最大化。因此，高度关联的机器应被包含在同一单元内，在这个过程中，定义为指派到每个单元的机器数量的单元规模以及单元数量就可以在设计过程中确定了。

表 9.2 零部件的加工计划和生产量

零部件	加工计划	生产量
1	$M_1 \to M_3 \to M_4$	5
2	$M_2 \to M_5 \to M_1$	10
3	$M_1 \to M_2$	2
4	$M_1 \to M_3 \to M_2$	7

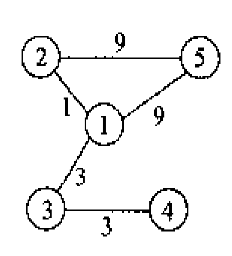


图 9.7 物流图

整数规划模型 在单元内加工的零部件或零部件族群使得与其他单元的交互联系最小化。显然，这等价于在单元内零部件加工总数的最大化。该模型基于在单元内零部件加工总数的最大化目标，对于机器制造元设计采用下列前提。

假设

1. 零部件族群通过聚类和编码构成
2. 每一零部件在加工计划单上都有一个预先确定的加工计划

符号

i,j——机器下标($i,j=1,2,\cdots,N_m$)；

k——零部件下标($k=1,2,\cdots,N_p$)；

c——单元索引($c=1,2,\cdots,N_c$)；

pv_k——零部件 k 的生产量；

cd_k——当使用移动装置时，零部件 k 在每一行程的有效转移单位；

U_c——单元规模的上限；

nt_{ijk}——零部件 k 在机器 i 和 j 之间的移动量：

$$nt_{ijk} = \left\lceil \frac{pv}{cd_k} \right\rceil,$$

其中$\lceil w \rceil$表示大于或等于 w 的最小整数值。

变量

$$x_{ic} = \begin{cases} 1, & \text{如果机器 } i \text{ 被指派到单元 } c \\ 0, & \text{否则} \end{cases}$$

该问题的整数规划模型可以构造如下：

$$\max \quad Z = \frac{1}{2}\sum_{c=1}^{N_c}\sum_{i=1}^{N_m}\sum_{j=1}^{N_m}\sum_{k=1}^{N_p} n t_{ijk} x_{ic} x_{jc} \tag{9.3}$$

$$\text{s.t.} \quad \sum_{c=1}^{N_c} x_{ic} = 1, \quad \forall i \tag{9.4}$$

$$1 \leqslant \sum_{i=1}^{N_m} x_{ic} \leqslant U_c, \quad \forall c \tag{9.5}$$

$$x_{ic} = \{0,1\}, \quad \forall i, c \tag{9.6}$$

目标函数(9.3)式使得在单元内加工处理零部件的总移动量最大化,结果是单元间的移动最小化,约束条件(9.4)确保所有机器都被分配于一个单元,约束条件(9.5)确保指派给每一单元的机器数量不超过单元规模的上限。约束条件(9.6)为二元变量限制。

遗传算法

遗传子表示和初始化 基于单元数的遗传子表示可以应用于这个问题,其解的染色体如下所示:

[2 1 3 1 2 3 2]

其中,染色体的长度取决于生产过程中机器的数量,每一个基因值表示已经指派了机器的单元编号。从而,我们可以知道在这个问题中有 7 台机器和 3 单元,机器 2,4 被指派到单元 1;机器 1,5,7 被指派到单元 2;机器 3,6 被指派到单元 3。遗传算法的第二步为染色体种群的初始化,初始化的过程可以随机进行或者由一组适应性好的种群构成。此处由随机方式产生初始种群。

评价 染色体通过适应值来评价,种群中的每一染色体都需计算其适应值,目标是寻找最大适应值的染色体。我们可以定义整数规划模型中的目标函数为适应值函数。对制造元设计问题,适应值可以简单地等价于如下的目标函数值:

$$\text{eval}(st_i) = z(st_i), \quad i = 1,2,\cdots,pop_size \tag{9.7}$$

需要说明的是,遗传算法可能会产生不满足约束条件的非法染色体,但是可以按如下程序进行修正:

修正程序

```
begin
for i = 1 to pop_size
    repeat
        检查每个染色体中的 U_c 和 N_c;
        if 不可行 then
            begin
                选择单元数溢出的机器;
                用随机产生的新单元数置换当前的单元数;
```

```
            更新数据;
        end
    until U_c 和 N_c 成为可行
end
```

遗传操作 为了繁衍新一代染色体(称之为子代),可以通过执行杂交和变异而形成。此处采用两分割点杂交(two-cut point crossover),例如,选择如下的染色体 st_1 和 st_2 进行杂交,随机选择的分割点位置在第 2 个和第 5 个基因后:

$$st_1 = [3\ 1 \mid 1\ 2\ 3 \mid 1\ 2]$$

$$st_2 = [2\ 3 \mid 2\ 1\ 1 \mid 2\ 1]$$

通过交换父代的左边和右边部分得到如下的子代:

$$op_1 = [2\ 3 \mid 1\ 2\ 3 \mid 2\ 1]$$

$$op_2 = [3\ 1 \mid 2\ 1\ 1 \mid 1\ 2]$$

杂交之后,进行随机变动的变异操作,随机选择一个基因并替换为一个由$[1,N_c]$内产生的单元编号数。如果染色体 st_1 的第 5 个基因被选中进行变异,产生的随机数是 1,经过变异后的染色体为:

$$op_3 = [3\ 1\ 1\ 2\ \underline{1}\ 1\ 2]$$

选择 选择策略是在种群空间中如何选择染色体。下一世代新种群的产生可以来自于所有父代和子代或者它们的一部分。制造元设计问题可以采用基于轮盘赌选择和最优性选择的混合选择策略。

数例 前一节提出的数值例子用来验证所提出的遗传算法。第一个例子有 8 台机器和 10 个零部件,每一单元的加工计划和生产量如表 9.3 所示。

表 9.3 8 台机器和 10 个零部件问题的数据

零部件	加工计划	生产量
1	$M_1 \to M_3 \to M_4$	1
2	$M_2 \to M_5 \to M_1$	1
3	$M_1 \to M_2$	3
4	$M_1 \to M_2 \to M_2$	1
5	$M_3 \to M_8 \to M_6$	1
6	$M_4 \to M_6 \to M_8 \to M_3$	4
7	$M_6 \to M_7 \to M_2$	2
8	$M_5 \to M_2$	1
9	$M_6 \to M_7 \to M_8$	1
10	$M_7 \to M_6 \to M_2$	2

假设生产中每一行程移动装置的有效容量是 1 个单位,机器间总零部件流量计算结果如表 9.4 所示。

表 9.4　两台机器间局部流

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_1	0	3	1	0	2	0	0	0
M_2		0	0	0	3	0	2	2
M_3			0	1	0	0	0	5
M_4				0	0	4	0	0
M_5					0	0	0	0
M_6						0	3	5
M_7							0	3
M_8								0

期望结果是安排 3 个单元，每个单元的规模为 3 或 4。为解决这个问题，采用上述的遗传算法，各项参数设置如下：最大迭代世代数：$max_gen-100$；种群规模：$pop_size=20$；杂交概率：$p_c=[0.5, 0.7]$；变异概率：$p_m=[0.1, 0.3]$；运行次数：10。在这个设置中，最好的杂交和变异概率是 $p_c=0.5$，$p_m=0.1$，对于所有 10 次计算运行，我们都获得了同样的结果。对于单元规模为 $U_c=4$ 的三个单元的例子，求得的最好染色体为[1 1 2 3 1 2 2 2]，即：机器 1,2,5 被指派到单元 1，机器 3,6,7,8 被指派到单元 2，机器 4 被指派到单元 3，适应值为 24，结果如表 9.5 所示。

表 9.5　制造元设计结果

单元数 3，单元规模 4	单元数 3，单元规模 3	
	方案 1	方案 2
单元 1：M_1 M_2 M_5 单元 2：M_3 M_6 M_7 M_8 单元 3：M_4	单元 1：M_4 M_6 M_7 单元 2：M_1 M_2 M_5 单元 3：M_3 M_8	单元 1：M_1 M_2 M_5 单元 2：M_3 M_4 单元 3：M_6 M_7 M_8
适值评价＝24	适值评价＝20	适值评价＝20
最大零部件流：34		

对大规模的例子，我们选择 46 个零部件和 18 台机器的问题。期望结果是安排 4 或 5 个单元，单元规模为 $U_c=5$ 或 6，每一零部件的生产量是常数 1。机器间零部件流量计算结果如表 9.6。

各项参数设置如下：最大迭代世代数：$max_gen=300$；种群规模：$pop_size=100$；杂交概率：$p_c=0.5$；变异概率：$p_m=0.1$，在这个数值例子中我们运行遗传算法 5 次，对于每一次运行结果，我们获得了最好的解决方案。最好的解决方案是单元数为 4，单元规模为 5，单元间零部件的最大流为 118，计算结果如表 9.7 所列。

表 9.6 燃气涡轮模型的机器间的零部件流

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}
M_1													11					
M_2								1	32						3			
M_3										3				1				
M_4						1	1											
M_5									1		16	8			16	23		
M_6							2		1									
M_7																		
M_8																		
M_9													4		6			
M_{10}																		
M_{11}																		
M_{12}																	8	
M_{13}																		
M_{14}																		
M_{15}																		
M_{16}																		
M_{17}																		8
M_{18}																		

表 9.7 制造元设计结果

单元数	单元规模	机器单元	适值评价	最大零部件流
4	5	单元 1：M_1 M_3 M_{10} M_{13} M_{14} 单元 2：M_2 M_5 M_9 M_{15} M_{15} 单元 3：M_4 M_6 M_7 M_8 单元 4：M_{11} M_{12} M_{17} M_{18}	116	146
	6	单元 1：M_1 M_2 M_8 M_9 M_{11} M_{13} 单元 2：M_4 单元 3：M_3 M_4 M_6 M_7 M_{10} 单元 4：M_5 M_{12} M_{15} M_{16} M_{17} M_{18}	118	
5	4	单元 1：M_5 M_{12} M_{15} M_{16} 单元 2：M_3 M_{10} M_{11} 单元 3：M_4 M_6 M_7 M_8 单元 4：M_1 M_2 M_9 M_{13} 单元 5：M_{14} M_{17} M_{18}	109	
	5	单元 1：M_3 M_8 M_{10} 单元 2：M_1 M_6 M_7 M_{11} M_{13} 单元 3：M_{14} 单元 4：M_4 M_{12} M_{17} M_{18} 单元 5：M_2 M_5 M_9 M_{15} M_{16}	113	

从实验结果看，提出的方法可以有效解决大规模的复杂制造元设计问题。

9.4.4 Joines的整数规划方法

为了克服基于次序的遗传算法及其他方法的局限性，Joines[332,333,336]发展了运用遗传算法的求解技术，解决了整数规划构造的制造元设计问题。

染色体表示 对于任何遗传算法，染色体表示需要描述种群中每一个个体。遗传子表示决定了遗传算法中问题是如何构成的，同时，也决定了所需采用的遗传操作。每一个个体或染色体是由一些符号组成的基因序列，这些符号可以是二进制数(0和1)，浮点数，整数，字母(如A,B,C,D)，矩阵，等等。文献[336]中的遗传子表示，前m个变量表示机器，后n个变量表示零部件，因此，每一个染色体是一个$m+n$整数变量的向量，取值范围从1到单元或族群的最大值($k_{\max}$)：

$$\text{染色体} \rightarrow (\underbrace{x_1, x_2, \cdots, x_m}_{\text{机器}}, \underbrace{y_1, y_2, \cdots, y_n}_{\text{零部件}})$$

在后面的9.5.2节，我们将认识到这种遗传子表示可以扩展到包含路径(或加工计划)的变量。

选择函数 通过对个体的选择不断地产生下一世代，这在遗传算法中起着极其重要的角色，基于染色体的适应值的随机选择，可以使得好的染色体被选中的机会增加。然而，种群中的所有个体都有机会被选中繁殖下一代，并且，种群中的染色体被选中的次数可以超过一次。选择操作有几种方案：例如：轮盘赌选择及其扩展形式、标度技术、锦标选择、最优性选择、排序选择[219,249,455]。

通常的选择方式是基于适应值，给每一个染色体j赋予一个选择概率P_j，产生N个随机数系列，并与种群的累积概率$C_i=\sum_{j=1}^{i} P_j$进行比较，如果$C_{i-1}\leqslant U(0,1)\leqslant C_i$，则选择相应的个体$i$复制进入新的种群。排序方法仅要求有一个评价函数可以将解影射到全排序集，可以允许极小化和负值，当所有的解排序后就按解的排序赋值P_i。规范化的几何等级[339]按下式定义每一个染色体的P_i：

$$P[\text{选择第}\ i\ \text{个染色体}] = q'(1-q)^{r-1} \tag{9.8}$$

其中，q是选中的最好染色体的概率，r是染色体的等级，1代表最好的，P是种群的规模。

$$q' = \frac{q}{1-(1-q)^P}$$

Joines[332,333,336]使用规范化的几何等级方案和最优性选择(上一代中最好的个体始终包含在当前的种群中)。

遗传算子 遗传算子确定了遗传算法的基本搜索机制。遗传操作是在基于种群中现有的解上产生新解。有两种基本的遗传算子：杂交和变异。变异算子倾向于在父代上作小的随机改变以形成一个子代，从而达到在整个状态空间的搜索。杂交算子包含了来自

父代的信息，组合产生两个子代，使得他们都从父代获得一定的相似(一组基因片断)。这两种算子基本类型的运用和派生物依赖于使用染色体表示形式。

Michalewicz[455]描述的6个浮点算子可以修正后用于整数的遗传子表示：均匀变异(uniform mutation)、多个均匀变异(multiuniform mutation)、非均匀变异(nonuniform mutation)、多个非均匀变异(multi-nonuniform mutation)、边界变异(boundary mutation)，简单杂交(simple crossover)，算术杂交(arithmetic crossover)[330]。基于提出的单元构造的遗传子表示，此处，提出两个问题特定的遗传算子(单元交换杂交(cell-swap crossover)，单元两点杂交(cell-two-point crossover))，提高了遗传算法的性能[336]。假设 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$ 和 $Y=(y_1,y_2,\cdots,y_n)$是两个 n 维整数行向量，代表来自于种群的个体(父代)。Michalewicz 描述的每一个算子在这个实验中都被应用，并简单讨论如下。定义 a_i 和 b_i 分别是变量 i 的下界和上界，对于单元构造问题(机器和零部件)，下界是0，上界是 $k_{\max}$(单元和族群的最大数)，P_j 为零部件与机器变量中零部件 j 的路径变量和路径数，这些算子如表9.8所示。父代从种群中随机地选择进行各种操作。

表9.8 在遗传算法中应用的参数

参　　数	参数值
边界变异算子数	4
均匀变异算子数	4
非均匀变异算子数	4
多个非均匀变异算子数	8
交换杂交算子数	6
多点杂交算子数	6
算术杂交算子数	6
单元的最大允许数 $k_{\max}$	k^L
最优个体的选择概率 q	0.08
最大世代数 $G_{\max}$	5000
种群规模	80

均匀变异：随机地选择一个变量 j，令其等于一个取尾的均匀随机数，$\lfloor U(a_i,b_i)\rfloor$，其中 $\lfloor x \rfloor$ 表示小于或等于 x 的最大整数。

$$x_i' = \begin{cases} \lfloor U(a_i,b_i) \rfloor, & \text{如果 } i=j \\ x_i, & \text{否则} \end{cases} \tag{9.9}$$

多个均匀变异：在父代中，对于所有变量运用(9.9)式。

非均匀变异：随机的选择一个变量 j，令其等于一个基于(9.10)式的非均匀随机数。新的变量等于老的变量加上或者减去一个随机置换。

$$x_i' = \begin{cases} \lceil x_i + (b_i - x_i) f(G) \rceil, & 如果\ r_1 < 0.5 \\ \lfloor x_i - (b_i + x_i) f(G) \rfloor, & 如果\ r_1 \geqslant 0.5 \\ x_i, & 否则 \end{cases} \tag{9.10}$$

其中

$$f(G) = [r_2(1 - G/G_{max})]^b \tag{9.11}$$

r_1 和 r_2 是 0 和 1 之间的随机数，G 是当前的世代，G_{max}是最大世代数，b 是形状参数。$\lceil x \rceil$ 是大于或等于 x 的最小整数。

多个非均匀变异：对于父代中的所有变量，运用(9.10)式。

边界变异：随机的选择一个变量 j，令其等于它的下界或上界，其中，$r=U(0,1)$。

$$x_i' = \begin{cases} a_i, 如果\ i = j, & r_1 < 0.5 \\ b_i, 如果\ i = j, & r_1 \geqslant 0.5 \\ x_i, 否则 \end{cases} \tag{9.12}$$

简单杂交：在 2 到 $m+n-1$ 的离散分布中产生一个随机数 r，根据以下规则产生两个新的个体 $\bar{X}'$和 $\bar{Y}'$：

$$x_i' = \begin{cases} x_i, & 如果\ i < r \\ y_i, & 否则 \end{cases} \tag{9.13}$$

$$y_i' = \begin{cases} y_i, & 如果\ i < r \\ x_i, & 否则 \end{cases} \tag{9.14}$$

算术杂交：算术杂交产生父代的两个互补线性结合，其中，$r=U(0,1)$。

$$\bar{X}' = r\bar{X} + (1-r)\bar{Y} \tag{9.15}$$

$$\bar{Y}' = (1-r)\bar{X} + r\bar{Y} \tag{9.16}$$

为了获取变量的整数表示，按如下执行：

$$\bar{X}' = (\langle ax_1 + by_1 \rangle, \langle ax_2 + by_2 \rangle, \langle ax_3 + by_3 \rangle, \langle ax_4 + by_4 \rangle, \langle ax_5 + by_5 \rangle) \tag{9.17}$$

其中

$$\langle ax_i + by_i \rangle = \begin{cases} \lceil ax_i + by_i \rceil, & 如果\ x_i > y_i \\ \lfloor ax_i + by_i \rfloor, & 否则 \end{cases}$$

单元交换杂交：上述遗传算子对任何整数规划都起作用。下面的两个算子仅对单元构造的遗传子表示起作用。令：$\bar{A}=(x_1, x_2, \cdots, x_m, y_1, y_2, \cdots, y_n)$和 $\bar{B}=(w_1, w_2, \cdots, w_m, z_1, z_2, \cdots, z_n)$为两个 $m+n$ 维单元构成的个体，按如下规则产生两个新的个体$\bar{A}'$和 $\bar{B}'$：

$$a_i' = \begin{cases} x_i, & 如果\ i < m \\ z_i, & 否则 \end{cases} \tag{9.18}$$

$$b_i' = \begin{cases} w_i, & 如果\ i < um \\ y_i, & 否则 \end{cases} \tag{9.19}$$

单元两点杂交：分别在 2 到 $m-1$ 间和 $m+2$ 到 $m+n-1$ 间的离散分布中产生两个

随机数，r_1 和 r_2，根据如下规则产生两个新的个体 $\overline{A}'$ 和 $\overline{B}'$：

$$a_i' = \begin{cases} x_i, & \text{如果 } i < r_1 \\ w_i, & \text{如果 } r_1 < i \leqslant m \\ y_i, & \text{如果 } m < i < r_2 \\ z_i, & \text{否则} \end{cases} \tag{9.20}$$

$$b_i' = \begin{cases} w_i, & \text{如果 } i < r_1 \\ x_i, & \text{如果 } r_1 < i \leqslant m \\ z_i, & \text{如果 } m < i < r_2 \\ y_i, & \text{否则} \end{cases} \tag{9.21}$$

Joines[336]证明了使用这些附加算子提高了算法的计算效率和解的精度。

种群的初始化 遗传算法必须有一个初始化的种群，最常用的方法是随机产生整个种群的各个解。然而，既然遗传算法能够迭代地改进现有的解(例如，来自于其他的启发式算法的解或当前的实际解)，潜在好的解的种子可以作为开始的种群，种群中的其他解由随机产生。Joines 描述的方法是随机地产生初始种群，在所有的方法中对每一复制都具有共同的随机数。

终止准则 遗传算法从一代迭代到下一代不断地选择和复制父代直到满足某一终止准则。最常用的终止准则是一个指定的最大进化世代数，另一个终止策略包含种群的收敛判断。通常，遗传算法迫使整个种群的多数收敛于单一的解。当种群中差异的总数小于指定的阀值时，算法就终止。也可以通过在一个指定的进化世代数内，其最好的解无改进时，算法就终止。或者，评价尺度的目标值能够建立在任意可接受的阀值之上，甚至，多个策略可以相互一起使用。

评价尺度 文献记载的几种数据集中，分组工效(grouping efficacy)的评价常作为评价标准来检验基于整数的遗传算法。由于分组工效在文献中经常应用并且其结果可用于比较分析，所以它可选为初始评价尺度，并且，产生块状对角线的单元构成和实际的可行解[381,483]。分组工效寻求在对角线块上的异常元素个数和无效元素(零)个数的最小化。异常元素表示零部件在单元间的移动，从而导致单元制造有效性的减少。更明确地说，分组工效企图使无效元素个数加上异常元素个数除以总操作域的最小化。操作域(operational zone)可以定义为操作数(异常元素加上沿着对角线的操作数)加上无效元素个数。当没有异常元素和无效元素时，分组工效的值为1；当异常元素个数等于操作总数时，分组工效的值为0。形式上，分组工效(Γ)定义如下：

$$\max \Gamma = (1-\psi)/(1-\varphi) = \frac{1-e_o/e}{1+e_v/e} \doteq (e-e_o)\ /\ (e+e_v) \tag{9.22}$$

其中 e 是数据矩阵中的操作数，e_v 是对角线块上的无效元素个数，e_o 是异常元素个数。

由于分组工效的非线性，不能用作通常的整数规划公式中的目标函数[336]。使用通

常的 0-1 型指派变量的说明和 9.4.3 节中的约束，以及分组工效(Γ)的定义，分组工效的非线近似形式如下：

$$\Gamma=\frac{\sum_{l=1}^{k_{\max}}\sum_{j=1}^{n}\sum_{i=1}^{m}y_{jl}x_{il}a_{ij}}{\sum_{j=1}^{n}\sum_{i=1}^{m}a_{ij}+\sum_{l=1}^{k_{\max}}\left[\left(\sum_{j=1}^{n}y_{jl}\right)\left(\sum_{i=1}^{m}x_{il}\right)\right]-\sum_{l=1}^{k_{\max}}\sum_{j=1}^{n}\sum_{i=1}^{m}y_{jl}x_{il}a_{ij}}$$

基于整数的遗传算法的检验 基于整数的遗传算法应用于文献上的 17 个数据集，检验其聚类的有效性。在所有的实验中，遗传算法应用基于规范化的几何分布排序策略和最优性策略，以及上述的7个基于整数的遗传算子。大量的数值检验后，Joines[336] 选择了表9.8中遗传算法的各参数值，这些参数表明对所有的数据集都运行得很好。这些值

族群/单元数	零 部 件	机 器
0	0,2,6,7,16	0,2,4,14,16,19,22,24,28
1	10,11,14,15,18	3,5,8,10,20,27,29,31,32,34
2	1,3,12,13,17	1,6,9,11,12,17,23,26,30
3	4,5,8,9,19	7,13,15,18,21,25,33

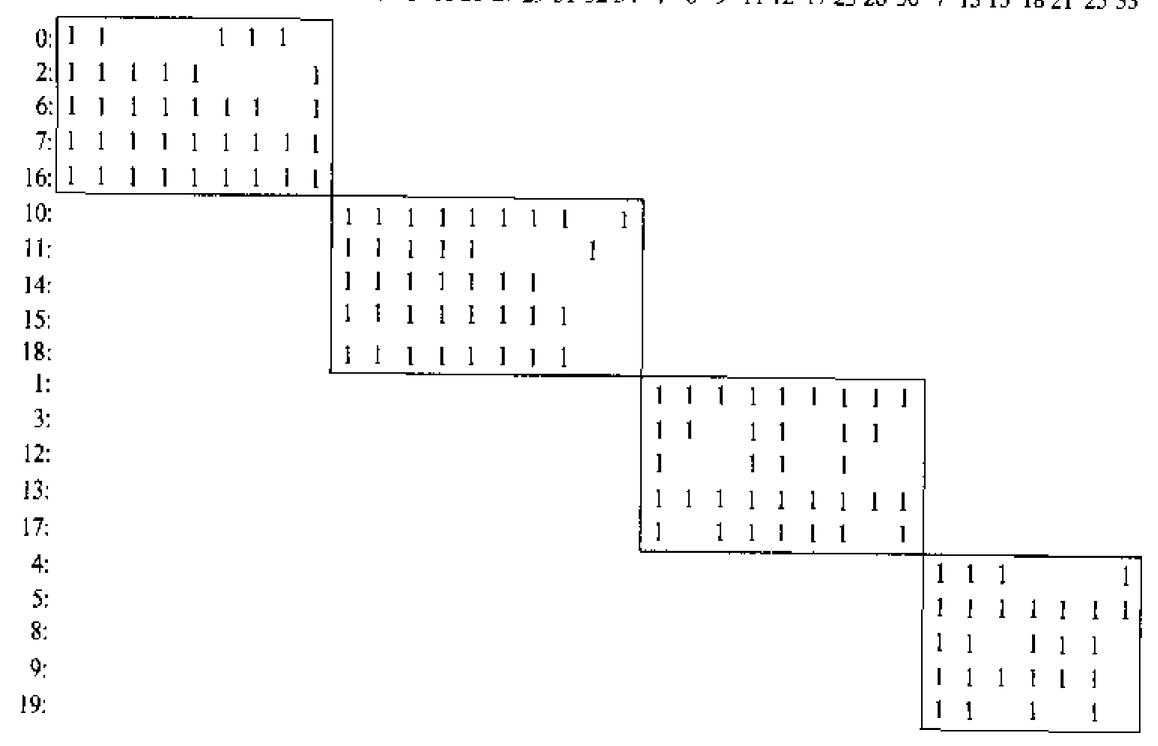


图 9.8 20×35 Burbidge 问题的整数遗传算法解(Γ=0.75751)

与各种算子联系表示了每一代中由那些算子所改进的父代个数。对于来自文献的数据集，最大允许单元值 k_{max} 最初设置为等于已知最好的单元数 k^b，如同被其他的单元构造算法所确定。由于选择分组工效作为准则来比较个体，结果强迫串沿着对角线块形成，并同时对零部件和机器分组。图 9.5 是 Burbidge[80] 的 20×35（零部件-机器）关联矩阵，该例子在 9.4.2 节的基于次序的检验中用过。这个数据集的解包含异常的元素，如图 9.8 所示。下一个检验的数据集是 King 和 Nakornchai[368] 的一个 16×43 的包含两个瓶颈机器的问题，解的结果如图 9.3 和 9.9 所示。需要注意的是，此时无需对输出结果作视觉检查确定机器单元和零部件族群组成，单元（族群）明确地由遗传算法确定，解的结果如图中的表格区域所示。

族群/单元数	零部件	机器
0	3,4,7,14	4,7,8,13,14,15,18,20,22,28,30,32,40,42
1	0,1,8,15	1,3,9,17,27,31,36,37,39,41
2	10,11,12	2,10,19,21,23,26,29
3	2,5,6,9,13	0,5,6,11,12,16,24,25,33,34,35,38

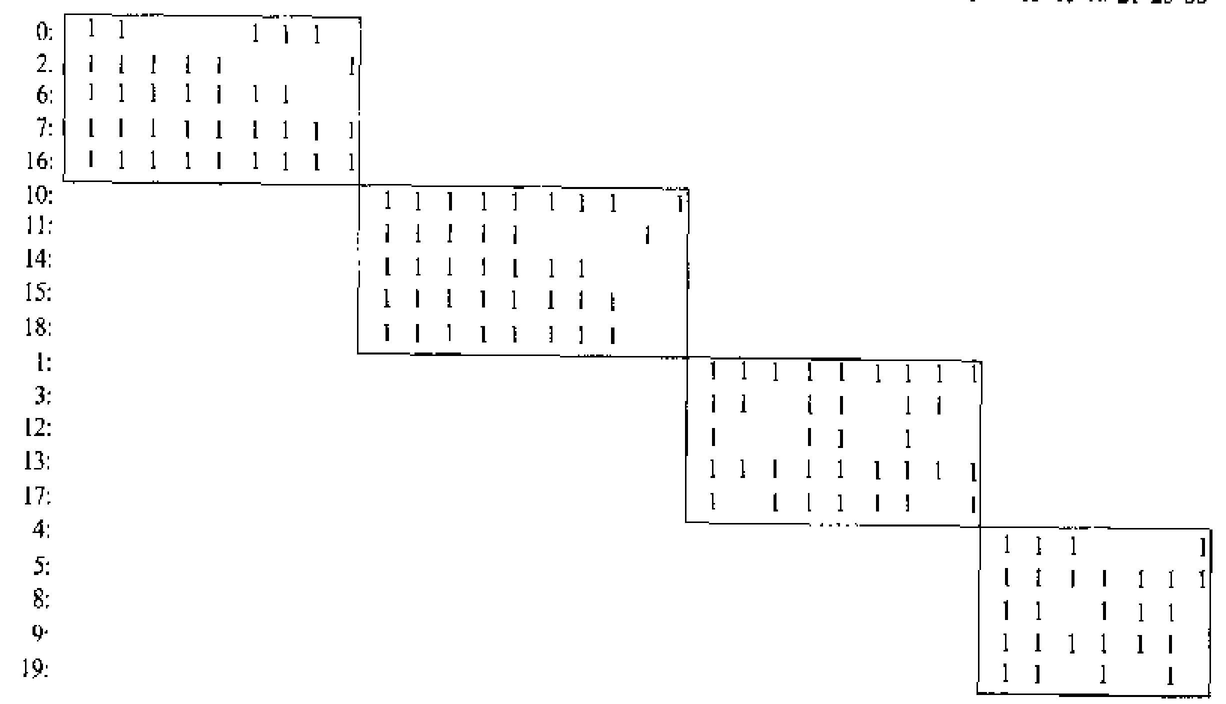


图 9.9 16×43 King 问题的整数遗传算法解（$\Gamma=0.4926$）

9.4.5 其他方法

使用基于次序的遗传算法通常需要对零部件-机器关联矩阵的视觉检查来确定单元族群的成分，在复杂的大规模的数据集中，这或许是非常困难的[333,335]。Billo，Tate 和

Bidanda[62,63]使用基于次序的遗传算法最小化单元数和最大化零部件的相似性。然而，他们的算法修改了遗传子表示，包含了单元间的分隔符（即，自动打断单元次序确定单元组成）。结果表明该方法优于层次型的聚类算法。Daskin[139]使用一种遗传算法确定一个好的初始机器排列，在排列中，机器基于它的次序而分成组，这使得在所有组中的机器的组合成本费用和单元间移动零部件的物料处理费用最小化。

Kazerooni，Luong 和 Abhary[350,351]发展了两种新的机器和零部件的相似系数，它包括生产量和加工顺序。和 9.4.2 节的方法相似，两个独立的基于次序的遗传算法分别被用来确定机器和零部件的最好次序，使得每一相似性最大化。

Venugopal 和 Narendran[640]使用遗传算法解决了多目标整数规划的单元构造问题，它使得单元间移动量和单元内的总荷载变化最小。这种算法仅仅使用 9.4.4 节中 Joines 使用的均衡变异和简单杂交，仅仅确定机器单元的组成。

9.5 可选加工计划的制造元设计

文献上的多数算法仅仅用零部件-机器关联矩阵，它对每一零部件仅仅包含一个单一固定的加工路径。相当少的研究者对可选操作、可选路径或在确定单元和族群时的机器冗余等问题进行研究。这些单一固定的路径经常被用于最优化功能性布局中机器利用的最大化，而不是单元制造收益的最大化。由于零部件可以按路径或计划安排到下一个可利用的机器，在功能上相似的机器和工作中心存在与否在功能性布局中就无需考虑了[21]。这些功能上相同的机器导致可选加工操作或路径，从而能够获得更好的制造元设计和独立性。

在 p-中值算法（p-median algorithm）中，Kusiak 和 Cho[384]拓展了一种数学规划方法，它包含完全固定的可选路径。Shtub[574]运用一般指派问题解决了类似的问题，试图获得更好的单元独立性。Nagi，Harhalakis 和 Proth[479]发展了一个两阶段方法，使得制造系统中的单元间的流量最小，同时考虑零部件的需求、容量、多重固定的零部件路径、多重功能相似的机器。Kang 和 Wemmerlov[345]发展了一个相似性下标的启发式算法，它能够在考虑零部件和机器意义上指定多重路径，首先确定机器单元，然后确定零部件族群。这种程序在将操作和路径结合时也考虑容量。然而，大多数的启发式方法在使用各种评价方法或者处理约束时缺乏单元形成的弹性。多数包括可选操作和多重路径的方法使用数学规划，但是限制了它们在解决大规模单元构造问题中的应用。

为了达到更满意的算法结果，应该包括其他的制造信息，例如：加工装置设置时间的要求、工具和工作人员的需求、机器容量、可选加工路径、机器费用、单元间的传递、单元内和单元间的布置、机器利用的简化等。

Joines[336]开发的制造元设计的整数规划模型，允许设计者替换各种类型的评价函数

参见9.4.4节，允许快速产生和回顾的可选路径的设计。Joines[332,335~338]等人通过结合新的遗传子表示和其他的制造信息（如，零部件顺序或路径信息），论证了遗传算法方法的弹性和可拓展性。

9.5.1 可选操作和机器冗余的结合

既然，当单元间存在移动时，单元制造的利益（简化在加工作业，生产时间等）将减少，所以，一个目标就是要形成完全独立的单元。多数的技术试图使单元间的移动最小化，它通常由异常元素的个数来表示。然而，异常元素个数或许不能精确地反映要求的单元间移动的水平。例如，如果在操作顺序的中间发生单元间的移动，将要求两个（不是一个）单元间的移动。操作顺序在识别单元间的流量和反向移动中是极有价值的。为了消除异常元素，引起单元移动的机器可以被重置，或者零部件路径顺序可以通过转包零部件、重新设计零部件或者使用可选路径或操作来改变。路径的产生经常使得机器利用最大化，而不是使得成组技术的效用最大化。由于对于一个可引起更大单元独立性的特殊操作，存在可选的机器，所以，可以限制零部件和一组特定的机器之间的固定关系[345]。

Chan和Milner[91]以及King[366]通过交互式地重置机器和重新求解问题，消除了异常元素。由于消除异常元素的行为会影响关联矩阵中其他的元素，所以异常元素是相互联系的。因此，在制造元设计程序之前过早地决定重置机器也许不会导致最好的解。其他学者建议在初始单元构成之后，通过交互式讨论重新设计零部件或将操作指派给另一台机器来减少单元数。由于实际问题大而复杂，这些方案都不太令人满意。

大多数的制造元设计算法使用一个二进制的零部件-机器矩阵$\mathbf{A}$，当零部件j需要在机器i上加工时，元素$a_{ij}=1$；否则$a_{ij}=0$。这种格式导致了没有机器置换的固定路径顺序。有些研究者已经提出了下列表示，其中，操作顺序包含在矩阵中[332,479]：

$$a_{ij}=\begin{cases}k, & \text{在机器 } i \text{ 上要求对零部件 } j \text{ 的第 } k \text{ 个操作}\\ 0, & \text{在机器 } i \text{ 上没有操作}\end{cases}$$

使用这种变量定义，通过指定能执行特定零部件的第k次操作的几个机器，可选操作可以融合进零部件-机器关联矩阵中（见图9.12）。因为评价函数独立于决定规则，遗传算法提供了相互交换各种目标函数的弹性而不改变算法。评价函数必须考虑新的零部件-机器关联矩阵表示。在使单元间的流最小化的前后关系中，指定的可选操作具有允许算法决定最合适路径顺序的优点。既然实际操作顺序是指定的，指定的可选操作也允许算法确定下列因素的实际影响：单元间移动、返回、顺序关联的设置时间等。

可选操作的整数规划模型 遗传算法包含可选操作的能力将通过使用9.4.4节的评价函数来表明。Joines[332,336]在9.4.4节使用的染色体表示并不改变，但是，分组工效的

计算必须允许可选操作。

对于构造成非线性整数规划的可选操作问题，需要在文献[336]定义的分组工效中加上下列新变量和约束。在这个模型中，一个特殊的操作也许会由一台以上的机器来执行。因此新变量 o_{jkl} 决定了选择哪一个操作。第一组约束确保了零部件 j 的操作 k 确切地被指派给一个单元，第二组约束确保了至少有一台可以执行操作 k 的机器被指派给同一个单元。

$$o_{jkl} = \begin{cases} 1, & \text{如果零部件 } j \text{ 的操作 } k \text{ 被指派于单元 } l \\ 0, & \text{否则} \end{cases}$$

$$\sum_{l=1}^{k_{\max}} o_{jkl} = 1, \quad j = 1,\cdots,n, \quad k = 1,\cdots,|O_j|$$

$$o_{jkl} \leqslant \sum_{i \in M_{kj}} x_{il}, \quad l = 1,\cdots,k_{\max}, \quad j = 1,\cdots,n, \quad k = 1,\cdots,|O_j|$$

其中，O_j 是零部件 j 的一组操作，M_{jk} 是对零部件 j 进行操作 k 的机器集合。使用这些新变量和定义的指派变量，x_{il} 和 y_{jl}，分组工效(Γ)可以定义如下：

$$\Gamma = \frac{\sum_{l=1}^{k_{\max}} \sum_{j=1}^{n} \sum_{k=1}^{|O_j|} y_{jl} o_{jkl}}{\sum_{j=1}^{n} |O_j| + \sum_{l=1}^{k_{\max}} \left[\left(\sum_{j=1}^{n} y_{jl} \right) \left(\sum_{i=1}^{m} x_{il} \right) \right] - \sum_{l=1}^{k_{\max}} \sum_{j=1}^{n} \sum_{k=1}^{|O_j|} y_{jl} o_{jkl}}$$

计算结果　注意到无论问题多么简单，都使用符号设置。只有评价函数必须对染色体解码。Nagi，Harhalakis，Proth[479]的例子用来证明遗传算法的能力，它使用这种允许可选操作替代的新的关联矩阵表示。这个问题由20个不同的零部件和20个工作中心组成。对所有的零部件，工作中心5和6与工作中心17，18和19一样可以相互交换。作为比较，修改Nagi的问题，移去所有重置的工作中心（机器6，18和19），可以产生一个如图9.10所示标准二元零部件和机器的关联矩阵。重复运行10次，遗传算法平均需要136.25进化世代数求解该问题。该问题的解如图9.11所示，$\Gamma=0.7308$ 和10个异常元素。对零部件-机器矩阵中使用操作顺序信息的进一步评价，如图9.12所示，显示了10个异常元素确实产生了13个单元间的移动。这说明了在精确地测量单元间的流量时，路径固定二进制表示的局限性。其次，遗传算法用来解决包含沿着操作顺序的机器冗余的问题（图9.12），重复运行10次，这种算法平均需要136.25进化世代数求得最终的 $\Gamma=0.7952$。图9.13的解仅包含一个异常元素，零部件4的第4个操作不能在它的单元4内加工，需要一个到其他3个单元（0，1或3）中的任何一个移动。因此，通过考虑挑选操作的可选机器和冗余的机器，遗传算法完成了对单元构成的改进。

零部件

机器	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0:		1	1	1	1															
1:						1		1	1	1										
2:											1	1	1							
3:																		1	1	1
4:						1	1			1										
5:	1	1	1	1		1	1	1	1	1										
7:											1	1	1							
8:	1		1		1															
9:														1	1	1	1			
10:											1		1							
11:	1	1	1	1	1															
12:																		1	1	
13:														1		1	1			
14:																		1	1	1
15:							1	1	1	1										
16:														1	1		1			
17:					1		1		1		1	1	1	1	1	1				

图 9.10　无机器冗余的原始矩阵

族群/单元数	零 部 件	机　器
0	5,6,7,8	1,4,5,15
1	10,11,12	2,7,10,17
2	0,1,2,3,4	0,8,11
3	13,14,15,16	9,13,16
4	17,18,19	3,12,14

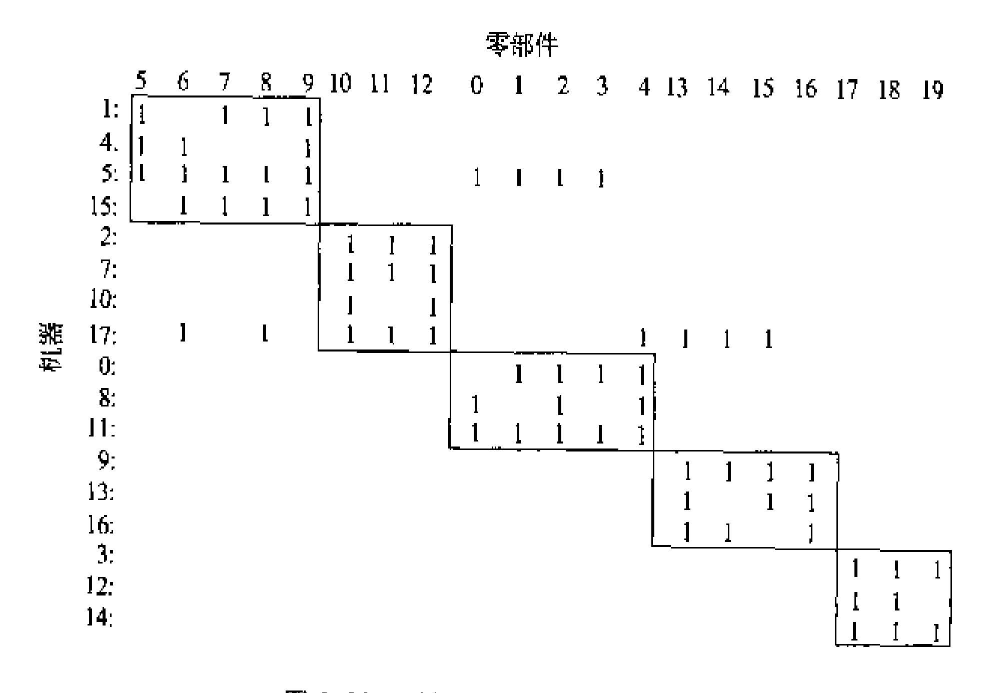


图 9.11　无机器冗余的 Nagi's 的解

机器 \ 零部件	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0:		3	1	1	2															
1:						3		3	2	1										
2:											2	1	3							
3:																		2	1	2
4:						2	1			3										
5:	3	1	4	3		1	4	2	4	4										
6:	3	1	4	3		1	4	2	4	4										
7:											1	2	2							
8:	2		2		1															
9:														1	3	2	3			
10:											3		1							
11:	1	2	3	2	3															
12:																		1	2	
13:														3		3	2			
14:																		3	3	1
15:							2	1	1	2										
16:														4	2		1			
17:					4		3		3		4	3	4	2	1	1				
18:					4		3		3		4	3	4	2	1	1				
19:					4		3		3		4	3	4	2	1	1				

图 9.12 零部件-机器操作原始关联矩阵

(来自于 Nagi 等人[479])

9.5.2 可选路径的结合

即使 9.5.1 节中的模型能够产生动态路径的单元，这个公式具有两个明显的缺点。首先，在聚类过程中，它是考虑可选操作不能被优先考虑的机器集指定和保护的前提下形成单元。这可能对拥有手动的机器和灵活的计算机化的数字控制系统(CNC)的制造商来说是重要的。举例来说，一个具有两个有效加工顺序的零部件，顺序之一使用手工操作的机器，顺序之二可能由单一的 CNC 工作站组成。9.5.1 节考虑的可选操作的遗传算法将允许 CNC 工作站被手工顺序中的任何一个单独的机器替代。实际上，一个完全可以在 CNC 工作站生产的零部件仅仅为了一个操作访问这个工作站，而其他操作由手工操作的机器完成，这种情况是不可能的。很有可能，那个零部件将会完全在 CNC 工作站中生产或单独使用手工操作的机器生产。这个模型在完全可选固定路径上的第二个缺点是无能力指定某些路径的优先权，而这些优先权可以允许制造元设计者产生平衡工作负荷的单元，达到最小的加工成本，充分利用加工技巧等。

固定可选路径的整数规划模型 克服上述两个缺点的完全固定可选路径(fixed alternative routings)的一般制造元设计问题可以由传统的整数规划模型来描述[346]，同时增加约束条件确保每一零部件被指派于 1 个且仅仅 1 个加工路径。然而，由于原始问题是一个 NP 完全问题[639,640]，很容易证明它也是一个 NP 完全问题。传统的公式必须结合以下附加的变量说明和约束条件来处理固定的路径：

$$z_{jh} = \begin{cases} 1, & \text{如果零部件 } j \text{ 使用从路径 } h \\ 0, & \text{否则} \end{cases} \tag{9.23}$$

$$\sum_{l=1}^{p_j} z_{jh} = 1, \quad j = 1, \cdots, m \tag{9.24}$$

其中，p_j 是零部件 j 的可选路径数。

族群/单元数	零 部 件	机 器
0	10,11,12	2,7,10,17
1	5,6,7,8,9	1,4,5,15,18
2	17,18,19	3,12,14
3	13,14,15,16	9,13,16,17
4	0,1,2,3,4	0,6,8,11

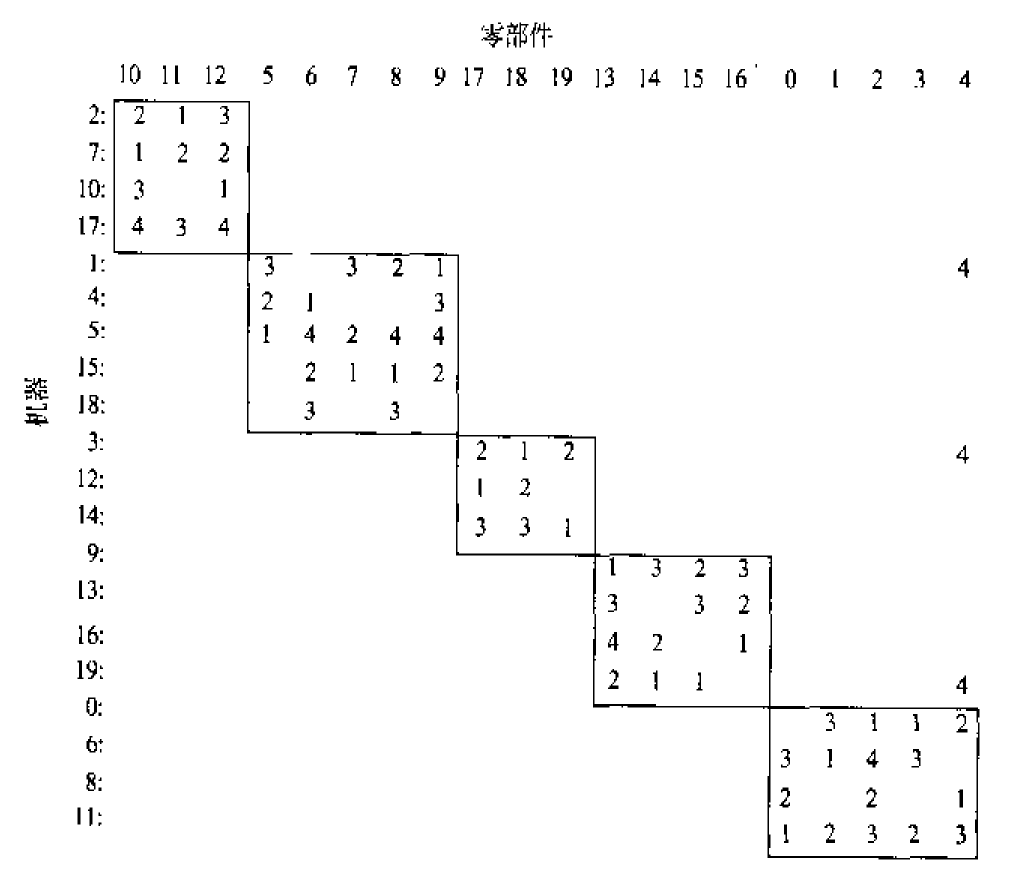


图 9.13 操作关联矩阵的解

（来自于 Nagi 等人[479]）

约束条件确保了每一零部件仅仅指派一个路径，非线性整数公式可以容易地按如下修正。使用这些新的变量和以前定义的变量，x_{il} 和 y_{jl}，分组工效(Γ)的定义如下：

$$\Gamma = \frac{\sum_{l=1}^{k_{max}}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{h=1}^{p_j} x_{il}y_{jl}z_{jh}a_{ijh}}{\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{h=1}^{p_j} z_{jh}a_{ijh} + \sum_{l=1}^{k_{max}}\left[\left(\sum_{j=1}^{n} y_{jl}\right)\left(\sum_{i=1}^{m} x_{il}\right)\right] - \sum_{l=1}^{k_{max}}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{h=1}^{p_j} x_{il}y_{jl}z_{jh}a_{ijh}}$$

通过一组新的允许固定可选路径的变量，遗传算法可以扩展为基于整数规划的遗传算法，特殊的，如果零部件 j 使用路径 h，令 $z_j=h$。如果零部件 j 仅仅只有单个路径，这个变量是多余的，因而不会包含在这个规划模型中。值得注意的是，此时的变量数已经增大，至多一个 n 因子。应用新的变量定义，通过对种群中的个体添加 z 变量分量，新的染色体表示可以是：

$$个体 \rightarrow (\underbrace{x_1, x_2, \cdots, x_m}_{机器}, \underbrace{y_1, y_2, \cdots, y_n}_{零部件}, \underbrace{z_1, z_2, \cdots, z_n}_{路径})$$

改进的分组功效和算法　再一次由于遗传算法的弹性，算法不需要改变，仅仅修改评价函数以适应这些新的遗传子表示。下面提供一个算法解释分组工效，以及个体的解码和分组工效的计算例子。

计算结果　遗传算法的新例子也是基于 Nagi 的问题。图 9.14 给出了每一个零部件都有可选加工路径的零部件-机器关联矩阵。注意到零部件 16 到 19 仅有一个加工计划，因此，遗传算法只要对 56 个不同的变量(20 台机器，20 个零部件，16 条路径分配)确定合

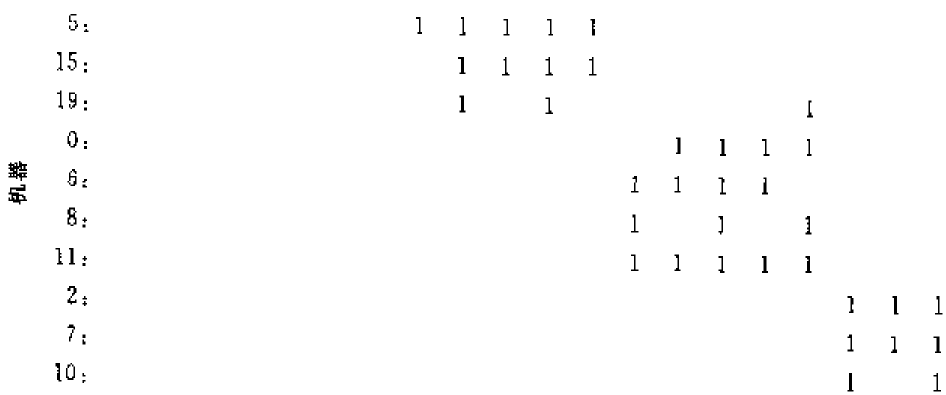
零部件:	P0		P1		P2		P3		P4			P5		P6						P7		P8						P9		P10			P11			P12			P13			P14			P15			P16	P17	P18	P19
路径:	1	2	1	2	1	2	1	2	1	2	3	1	2	1	2	3	4	5	6	1	2	1	2	3	4	5	6	1	2	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	1	1	1
0:			1	1	1	1	1	1	1	1	1																																								
1:												1	1							1	1	1	1	1	1	1	1	1	1																						
2:																														1	1	1	1	1	1	1	1	1													
3:																																																	1	1	1
4:												1	1	1	1	1	1	1	1									1	1																						
5:	1		1		1		1					1					1	1	1		1		1		1		1	1																							
6:		1		1		1		1					1	1	1	1				1		1		1		1			1																						
7:																														1	1	1	1	1	1	1	1	1													
8:	1	1			1	1			1	1	1																																								
9:																																							1	1	1	1	1	1	1	1	1	1			
10:																														1	1	1				1	1	1													
机器 11:	1	1	1	1	1	1	1	1	1	1	1																																						1	1	
12:																																							1	1	1				1	1	1	1			
13:																																																	1	1	1
14:														1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1																						
15:																																							1	1	1	1	1	1				1			
16:										1					1			1						1	1					1			1			1			1			1			1						
17:									1					1			1					1	1								1			1			1			1			1			1					
18:											1					1			1							1	1					1			1			1			1			1			1				
19:																																																			

图 9.14　具有完全可选路径 Nagi 问题的原始矩阵

适的值。和先前的模型一样,仅有一个异常元素和单元间的移动(如图 9.15 所示结果中的零部件 4)。在平均 231.6 的进化世代数之后,获得同样的评价值 $\Gamma=0.7952$。在这个问题中,由于完全可选路径矩阵能够简化到如图 9.12 的操作顺序矩阵,所以获得了相同的单元构成。如果每一路径的操作数是不同的,这将是不可能的,例如,如果零部件 1 能够在机器 5,8,11 或 6,9 上生产,这两个可选加工路径不能够简化到如图 9.3 的操作顺序矩阵的形式。因此,形成不同的单元结构是可能的。

族群/单元数	零部件	机器
0	13,14,15,16	9,13,16,17
1	17,18,19	3,12,14
2	5,6,7,8,9	1,4,5,15,19
3	0,1,2,3,4	0,6,8,11
4	10,11,12	2,7,10,18

零部件	0	1	2	3	4	5	6	7	8	9
路径	2	2	2	2	3	1	6	2	6	1
零部件	10	11	12	13	14	15	16	17	18	19
路径	2	2	2	1	1	1	—	—	—	—



机器 \ 零部件	13	14	15	16	17	18	19	5	6	7	8	9	0	1	2	3	4	10	11	12
9:	1	1	1	1																
13:	1		1	1																
16:	1	1		1																
17:	1	1	1	1																
3:					1	1	1													
12:					1	1														
14:					1	1	1													
1:								1		1	1	1								
4:								1	1			1								
5:								1	1	1	1	1								
15:									1	1	1	1								
19:									1		1						1			
0:														1	1	1	1			
6:													1	1	1	1				
8:													1		1		1			
11:													1	1	1	1	1			
2:																		1	1	1
7:																		1	1	1
10:																		1		1
18:																		1	1	1

图 9.15 具有完全可选路径 Nagi 问题的解

带有机器冗余的完全可选路径 即使考虑完全可选路径(complete alternative routings),它提供了指定一组偏好的机器或对路径指派优先权的能力,这仍然是有局限性的。操作顺序模型提供了允许冗余机器和确定单元间移动的真实影响的优点,通过允许完全可选路径包含它们的操作顺序,这两个模型可以合并成一个表示。

再一次由于遗传算法的弹性,算法不需改变,仅仅修改评价函数以适应这些带有完全可选路径的操作顺序的结合。这个遗传算法来自于对 Nagi 问题的修改,图 9.16 中给出了每一零部件都有可选加工路径的原始零部件-机器关联矩阵,现在,每一个可选路径包含了操作顺序。一些路径包含了冗余的机器(例如,一个路径的某些操作可以在几个不同的机器上运行)。与前面相同,零部件 16 到 19 仅有一条路径。和 9.5.2 节一样遗传算法仅仅需对 56 个变量确定合适的值。如图 9.17 所示,这个模型能够消除异常的元素,平均 227.7 的进化世代数获得略高的评价值 $\Gamma=0.7975$。值得注意的是,遗传算法能够确定最合适使用的加工路径和动态地构建操作顺序。

零部件:	P0		P1		P2		P3		P4			P5		P6				P7		P8			
路径:	1	2	1	2	1	2	1	2	1	2	3	1	2	1	2	3	4	1	2	1	2	3	4
0:			3			1		1			2				1								
1:													3					1	3				2
2:		1			2		3									1						1	
3:					1				2														
4:													2				1						
5:	3		1					3					1		2		4		2		2		4
6:	3		1			4		3					1		2		4		2		2		4
7:					1											2						2	
8:	2					2					1												
9:										2				1						2			
10:																							
11:	1		2			3		2			3										1		
机器 12:				2			2											2		1			
13:										1		1		2									
14:					3		1		1									3					
15:																	2		1				1
16:												2		3						3			
17:		2			3					3	4	3				3	3					3	3
18:		2			3					3	4	3				3	3					3	3
19:		2			3					3	4	3				3	3					3	3

零部件:	P9		P10		P11		P12		P13		P14		P15		P16	P17	P18	P19
路径:	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	1	1	1
0:		1																
1:			1				1											
2:				2		1		3	1									
3:			2		2											2	1	2
4:		3					2				3	4						
5:		4	3				3											
6:		4	3				3											
7:				1		2		2	3									
8:																		
9:	1									1		3		2	3			
10:				3				1	4									
11:																		
机器 12:																1	2	
13:	2									3				3	2			
14:																3	3	1
15:		2			1						2		2					
16:	3									4		2			1			
17:				4		3	4	4	2	2	1	1	1	1				
18:				4		3	4	4	2	2	1	1	1	1				
19:				4		3	4	4	2	2	1	1	1	1				

图 9.16 具有完全可选路径和机器冗余的原始矩阵

通过将机器冗余和完全固定的可选加工路径的结合,这个模型提供了比以前的模型明显的优点。尽管,前一节的模型通过使用单独的加工路径可以处理机器冗余,这一节的方法在计算效率和状态空间方面上显得效率更高。举同样例子,一个零部件可以在单一的 CNC 工作站上加工或者在一组手工操作的机器上加工。如前所述,如果不想让遗传算法指派某些操作到 CNC 工作站,指派剩余操作于手工操作的机器,因而,需要指定一组偏爱的机器。然而,一些手工操作的机器可以相互替换或者可以指派给一组可互换的多目的 CNC 机器。遗传算法具有这种性能,在最合适的完全可选路径中,它使用最合适的 CNC 工作站或一列手工操作的机器。同样,这个模型与完全可选路径模型不同,它能够确定单元间移动的实际影响。

族群/单元数	零 部 件	机 器
0	7,8,12	1,4,5,15,19
1	0,3	0,6,8,11
2	1,17,18,19	3,12,14
4	4,5,9,13,14,15,16	9,13,16,18
3	2,6,10,11	2,7,10,17

零部件	0	1	2	3	4	5	6	7	8	9
路径	1	1	1	1	2	1	3	2	4	2
零部件	10	11	12	13	14	15	16	17	18	19
路径	2	2	2	2	2	2	—	—	—	—

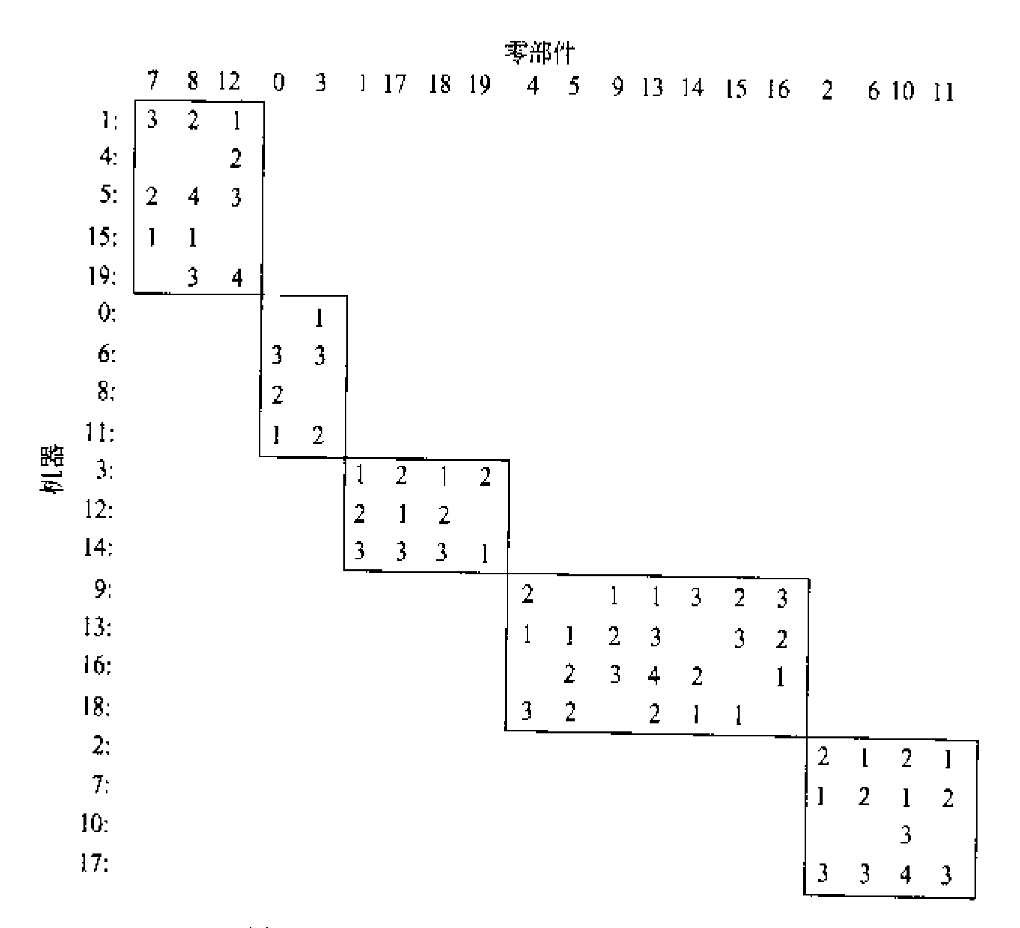


图 9.17 具有完全可选路径 Nagi 问题的解

9.5.3 Moon,Gen 和 Kim 的对于独立单元的方法

异常的元素(单元间的移动)降低了制造单元的有效性。因此,多数的方法尽量减少异常元素的个数。在许多案例中,在当前条件下(零部件,机器,加工计划等),要消除所有的异常元素是不可能的。Moon 和 Kim[462]考虑了一个类似于 Joines[332,336]的利用可选固定路径的更为复杂的模型,但是需要的时候可以通过重置机器,产生完全独立的制造单元。这个目标是使生产成本和重置机器的成本最小化。

问题描述 在这一节,提出一个能同时确定零部件、零部件族群和机器单元的 0-1 型整数规划模型,它结合了加工计划和其他的制造因素。

假设

1. 已知每一个零部件的可选加工计划
2. 已知结构的单元数和每一单元中的零部件数的上限
3. 在每一个单元中仅存在同种类型的机器

符号说明

i——零部件下标($i=1,2,\cdots,np$);
j——加工计划下标($j=1,2,\cdots,npp_i$);
k——机器下标($k=1,2,\cdots,nm$);
c——单元下标($c=1,2,\cdots,nc$);
p_{ij}——零部件 i 的加工计划 j;
ms_{ij}——当选择加工计划 j 时对零部件 i 的一组机器;
pm_{ij}——集的势 ms_{ij};
pc_{ijk}——在加工计划 j 下机器 k 对零部件 i 的加工成本;
mc_k——机器类型 k 的成本;
U_c——单元 c 中零部件数的上限;
um_k——机器类型 k 的重置数;
cam_k——机器类型 k 的能力;
pv_i——零部件 i 的生产量;

$$x_{ijc}=\begin{cases}1, & \text{如果在加工计划 } j \text{ 下生产的零部件 } i \text{ 属于单元 } c\\ 0, & \text{否则}\end{cases}$$

$$y_{kc}=\begin{cases}1, & \text{如果机器类型 } k \text{ 属于单元 } c\\ 0, & \text{否则}\end{cases}$$

模型的目标函数是使得机器加工和机器重置成本的和最小,其模型可以构造如下:

$$\min \quad TC=\sum_i\sum_j\sum_k\sum_c pv_i\, pc_{ijk}\, x_{ijc}+\sum_c\sum_k mc_k y_{kc} \tag{9.25}$$

$$\text{s.t.} \quad \sum_j\sum_k x_{ijc}=1, \qquad \forall i \tag{9.26}$$

$$\sum_i\sum_j x_{ijc}\leqslant U_c, \qquad \forall c \tag{9.27}$$

$$\sum_i\sum_j pv_i pc_{ijk}x_{ijc}\leqslant cam_k, \quad \forall (k,c) \tag{9.28}$$

$$\sum_{k\in ms_{ij}} y_{kc}\geqslant pm_{ij}x_{ijc}, \qquad \forall (i,j,c) \tag{9.29}$$

$$x_{ijc},y_{kc}=\{0,1\}, \qquad \forall (i,j,k,c) \tag{9.30}$$

约束条件(9.26)确保对每个零部件仅仅选择一个加工计划,每个零部件仅仅属于一个单

元；约束条件(9.27)强加了一个单元中零部件最大数的限制，便于计划和控制；约束条件(9.28)表示一种机器类型的总加工费用小于或等于它的有效容量；约束条件(9.29)表示在选定的操作计划下零部件的生产而需要的所有机器应被指派于同一个单元；约束条件(9.30)表示 0-1 型的整数变量。

遗传算法 上述模型有一些局限性。由于变量被限于整数值，由于计算的复杂性，这个模型很难求解机器和零部件规模大的问题。同样，这个模型也不能给制造元设计者提供弹性来改变目标函数和约束条件。在这一章中，使用遗传算法的方法来克服这些缺点，来解决同时进行加工计划的选择和独立制造元设计的问题。

遗传子表示和初始化 遗传子表示是制造元设计问题运用遗传算法的第一步。在机器制造元设计问题中，每一个基因表示加工计划编号和每一零部件的单元编号，染色体的长度表示所考虑问题的零部件个数。假设零部件的个数为 np；那么染色体的表示如下：

$$S_i = [m_1 m_2 m_3 \cdots m_k \cdots m_{np}]$$

其中 S_i 是第 i 个染色体，m_k 是染色体 S_i 中的第 k 个基因。在这个染色体中，令每一零部件的加工计划数为 npp_k，单元数的上限为 cn，m_k 的取值范围为$[1, npp_k \times cn]$。如果 d 是来自 S_i 的一个 m_k，对零部件 i 的加工计划编号和单元编号可以计算如下：

$$j \leftarrow (d-1) \ / \ (cn+1) \tag{9.31}$$

$$c \leftarrow (d-1) \bmod cn + 1$$

由等式(9.31)，S_i 可以表示如下：

$$SS_i = [(j_1, c_1), (j_2, c_2), (j_3, c_3), \cdots, (j_k, c_k), \cdots, (j_{np}, c_{np})]$$

例如，令 $np=5, npp_1=2, npp_2=3, npp_3=2, npp_4=1, npp_5=2, cn=4$，可以得到如下的染色体：

$$S_1 = [5\ 2\ 7\ 1\ 6]$$

由等式(9.31)，S_1 对每一零部件计算加工计划和单元数，上述染色体计算结果如下：

$$SS_1 = [(2,1)\ (1,2)\ (2,3)\ (1,1)\ (2,2)]$$

遗传算法的第二步是初始化染色体的种群。在这个模型中，染色体随机产生。

评价 在许多最优化问题中，目标函数很自然地定义为成本函数的最小化。制造元设计问题的目标函数也是构造成使机器加工和重置成本最小化。其适应值的计算基于初始的目标函数。染色体 i 的适应值评价定义如下：

$$fit_fun\ (i) = TC \tag{9.32}$$

然而，制造元设计问题的解决空间包含两部分：一个是可行区域，另一个是不可行区域。为了处理不可行的染色体，我们使用再生技术。这种技术对不可行的解由两步构成，如果一个染色体不在可行域内，首先，丢弃它；其次，再产生一个新的染色体。

杂交和变异 杂交的目标在于交换两个父代的字符串。在这个模型中使用的是单分

割点杂交方法。例如，随机选择两个父代的染色体，分割点随机的选择在位置2，如下所示：

分割点

↓

[5 2 7 1 6]

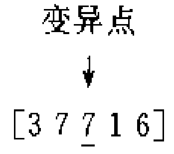
[3 7 2 2 5]

通过交换父代染色体的末端，形成的子代如下：

[5 2 2 2 5]

[3 7 7 1 6]

在一定的变异率下，染色体的每一个值被随机地选择进行变异。制造元设计问题的变异操作按随机交换进行。选择一个基因 m_k，被$[1, npp_k \times xcn]$内的随机整数替换。如下例：



在这个染色体中，第3个基因被选择中进行变异，其基因的值被[1,12]内的随机整数值替换。如果这个随机整数值是4，变异后的子代为：

[3 7 4 1 6]

选择　选择策略采用确定性的选择策略。例如，按升序排列的父代和子代，删除所有的重复个体，选择种群中前 *pop_size* 个染色体作为新的种群。

数值例子　为了说明所讨论的方法的有效性，假设一个制造系统有10台机器，7个零部件，3个单元，每一台机器的重置成本为200，每一单元的最大零部件限制数为3。一组加工计划、每一零部件的生产量、每单位的加工成本以及机器容量如表9.9所示。

表9.9　零部件-机器加工的数据

零部件	*npp*	机器										pv_i
		1	2	3	4	5	6	7	8	9	10	
1	1	2		3							2	80
2	1				2					3	1	80
	2	4		5		4				7		
	3	6		5		7				5		
3	1	6				5		6				80
	2					3		3				
4	1			4		5					7	80
	2		4			7					6	
	3					3				4	3	

续表

零部件	npp	机器										pv_i
		1	2	3	4	5	6	7	8	9	10	
5	1			4	2					5	2	80
6	1	5	6				4		8			80
	2	2	4				2			4		
7	1			5	7					3		80
		2500	2300	2000	2200	2000	2500	2500	2000	2000	2000	

模型的遗传操作参数如下：$p_c=0.6$，$p_m=0.2$，$max_gen=1000$，$pop_size=100$。这个计算实验在奔腾处理器 75 芯片和 Windows 95 操作系统环境下的个人计算机上进行。在这种参数设置下，运行遗传算法 5 次，获得了两个可选的解，由表 9.10 给出。

表 9.10 最佳的解决方案

解	单元数	零部件(计划)	机　器	适应值	CPU 时间/s
方案 1	1	1(1)5(1)7(1)	1 3 4 9 10		
	2	2(1)3(2)4(3)	5 7 9 10		
	3	6(2)	1 2 6 9	8120	247
方案 2	1	3(2)	5 7		
	2	2(1)4(3)6(2)	1 2 5 6 9 10		
	3	1(1)5(1)7(1)	1 3 4 9 10		

同样使用 LINDO/PC 软件包，求解 0-1 型整数规划模型，运算 5 次得到的最优解为 8120，平均计算时间是 340s。为了评价基于遗传算法方法的有效性，采用了 3 种不同规模的问题，最好的计算结果如表 9.11 所示。

表 9.11 结果比较

问题规模 $m\times p$	遗传算法的 CPU 时间/s	LINDO 的 CPU 时间/s	最佳的值
10×10	293	440	10,720
10×15	362	549	13,780
10×20	637	不能求解	18,438

在表 9.11 表中，基于遗传算法的方法能够在满意的计算时间内找到最优解。然而，LINDO 方法由于约束条件和整数变量的数量变大了，不能解决第 3 个问题。

9.6 独立单元的设计

在这一节，将讨论遗传算法在独立制造单元设计的应用潜力。所谓独立单元是指产品在一个单元内完成加工而无需访问其他任何单元。这种现象在劳动密集型的制造单元是常见的，此时，往往是产品的重量轻、费用低、使用小型的机器和设备。由于机器费用不高，机器的重置就不会在机器密集单元内构成主要关心的因素。因此，具有独立单元的主要原因之一就是简化加工计划和工序过程。工序任务越简单，完成工序的可能性也就越大。进而，由于高的生产量，将多个单元指派给一个零部件族群就非常普通了。

除了劳动密集单元，独立单元大概是有些规定环境中惟一允许的单元类型，在这种场合，由于加工跟踪的需要，产品不允许离开本单元与其他单元共享机器。这也阻止了产品间的相互混合，对于制药和医药装置的工业来说，这是非常关键的。因此，识别产品族群在这种场合是非常重要的。独立单元情况下的制造元设计可以看成是一个排序过程，首先，确定产品族群，然后，形成相应的独立单元。

9.6.1 机器类型数最小化的族群构造

本节的目的是要构造产品族群，以便最小化机器类型的总数。

遗传子表示 Süer 等人[597]提出了一种基于族群数的遗传子表示来描述该问题。基于族群数的遗传子表示确保每一个零部件指派给一个族群，X_i 表示指派给产品 i 的族群编号，族群的个数 f 是设计本身的问题，它可以由一个族群内期望的产品数或生产组织来确定。举例来说，以下是代表这种解的一个染色体：

[1 2 3 1 2 3]

其中，染色体的长度表示在制造元设计中需要考虑的产品数。该例子说明 6 个零部件需要指派到 3 个族群。零部件 1 和 4 指派给族群 1，零部件 2 和 5 指派给族群 2，零部件 3 和 6 指派给族群 3。

一个零部件指派给任何族群的概率是相同的，它可以是 $1/f$。零部件到族群的指派关系可以通过随机数来确定相应的族群进行。然而，也有可能一些族群没有零部件指派给它们，这将导致比初试计划少的族群数。况且，较少的族群数导致较少的机器类型总数，从适应值的角度来说，这就显得不太合适了。因此，需要修正一个过程避免这种情况的发生，即，仔细地填满染色体中的最后$(f-1)$个位置确保至少有一个零部件指派给每一个族群。

适应值函数：此处使用的适应值函数是需要的机器类型的总数。第一步是确定每个族群需要的机器类型数，它由指派给族群 k 的所有产品的机器集合的并确定，即：

$$\mathrm{SM}_k = \bigcup_{i \in \mathrm{SP}_k} \mathrm{SO}_i \tag{9.33}$$

其中，SO_i 是产品 i 的机器类型集合，SP_k 是包含在族群 k 内的产品集合，SM_k 是族群 k 需要的机器类型的集合。

确定了每个族群需要的机器类型的集合，从 SM_k 就能容易地求出每个族群(NM_k)机器类型数。下一步就是求出所有族群(FF)的机器类型总数，即：

$$FF = \sum_{k=1}^{f} NM_k \tag{9.34}$$

选择操作 在 Süer 等人的研究中，选择了轮盘赌选择操作。每个染色体基于它的适应值赋予一个复制概率。由于目标是极小化的形式，适应值越小，复制概率就越大。因此，需要有个转换函数。一个可选的方法是首先求所有适应值的总和，然后，通过每一染色体相应的适应值除以总适应值来确定该染色体调整后的适应值。最后，复制概率由式(9.37)显示的调整后的适应值来确定。

$$TFF = \sum_{i=1} FF_i \tag{9.35}$$

$$AF_i = TFF \;/\; FF_k \tag{9.36}$$

$$P_i = AF_i \Big/ \sum_{i=1}^{s} AF_i \tag{9.37}$$

其中，FF_i 为染色体 i 的适应值，TFF 是总适应值，AF_i 是染色体 i 调整后的适应值，P_i 是染色体 i 的复制概率，S 是染色体总数。

杂交算子 此处采用单分割点杂交。随机选定分割点，然后将分割点后两个父代的基因交换，形成子代。在下例中，假设分割点在第 4 个基因，第 1 个父代的最后两个基因被第 2 个父代的最后两个基因替换，即：

父代 1 [1 2 3 1 3 1]
父代 2 [2 1 3 2 1 2]
子代 1 [1 2 3 1 1 2]
子代 2 [2 1 3 2 3 1]

杂交的结果，可能产生不是每一个族群都至少有一个零部件指派的染色体，因此，需要进行可行性检查。在这种情况，可以随机选定一个零部件指派给空缺的族群。

变异算子 变异操作是独立地应用于每一个基因直到有一个变异，一旦有一个基因变异了，变异操作就终止。当一个基因选中进行变异，产品的位置仍旧不变，惟一的变化是族群的指派关系。在以下的例子中，变异发生在第 4 个基因，而产品 4 指派给族群 3。在这个问题中，变异操作是非常需要的，主要原因是初始种群中的染色体期望每一个族群有相等的零部件(以 $1/f$ 的概率指派零部件给族群)。由于族群是按加工的相似性形成的，而不是按零部件数相等形成的，这个初始分组有可能不是最期望的族群。希望杂交和变异能够纠正这个问题。与杂交的情况相同，某个族群中的惟一一个零部件选种进行变

异，结果造成空的族群，因此，需要进行可行性检查。在这种情况可以随机选定一个零部件再指派给该族群的空确位。

$$\text{原染色体}\quad [1\ 2\ 3\ \underline{1}\ 2\ 3]$$

$$\text{变异后的染色体}\quad [1\ 2\ 3\ \underline{3}\ 2\ 3]$$

杂交与变异策略 Süer 等人[594]采用如下的杂交和变异策略：

1. 杂交和变异($C+M$)：成对地进行染色体的复制，并且都参加杂交操作。杂交操作后，每个基因都单独检测直到有一个基因进行变异。一旦，有一个基因变异，其他基因就无需再检测进行变异。
2. 单一杂交(C-O)：这一策略与上述相似，只是省略变异。
3. 单一变异(M-O)：染色体的复制只由变异进行，在这种情况，随着变异概率的增加，变异可以发生在每一个基因。第1个基因的变异概率为$1/n$，第2个基因的变异概率为$1/(n-1)$，第3个基因的变异概率为$1/(n-2)$，依次类推。
4. 变异和杂交(M_x,C_y)：被复制的染色体中x条进行变异操作，y条染色体进行杂交操作。采用的变异策略是以(M-O)概率递增的形式进行。

数值例子：表9.12的产品-机器关联矩阵用于说明工作程序，在矩阵中，$M_{ij}=1$表示零部件j在机器i上加工，$M_{ij}=0$表示零部件j不在机器i上加工。

表 9.12 机器-零部件关联矩阵

机器	产品									
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
M_1	1			1	1			1		
M_2				1	1		1	1	1	1
M_3	1		1	1		1	1	1	1	1
M_4	1	1	1	1	1	1	1		1	
M_5		1				1				
M_6		1	1			1				

假定种群由以下染色体组成：

$$V_1=[1\ 2\ 1\ 3\ 1\ 2\ 2\ 3\ 1\ 3]$$

$$V_2=[1\ 2\ 3\ 3\ 1\ 3\ 2\ 2\ 1\ 2]$$

$$V_3=[2\ 3\ 1\ 3\ 2\ 1\ 2\ 2\ 1\ 3]$$

$$V_4=[3\ 1\ 2\ 2\ 2\ 1\ 3\ 1\ 3\ 2]$$

计算的详细过程由第1条染色体来说明。随机产生3个族群，族群1由产品1,3,5,9组成，族群2由产品2,6,7组成，族群3由产品4,8,10组成。族群的分组情况为：

$$\mathrm{SP}_1=\{P_1,P_3,P_5,P_9\}$$

$$\mathrm{SP}_2=\{P_2,P_6,P_7\}$$

$$\mathrm{SP}_3=\{P_4,P_8,P_{10}\}$$

利用表 9.12 的关联矩阵，很容易确定机器加工集合，例如：零部件 1 需要机器 1,3,4，第 1 族群所有零部件的机器集合归纳如下：

$$SO_1 = \{M_1, M_3, M_4\}$$
$$SO_2 = \{M_3, M_4, M_6\}$$
$$SO_3 = \{M_1, M_2, M_4\}$$
$$SO_4 = \{M_2, M_3, M_4\}$$

因而，族群 1 的机器集合确定为：

$$SM_1 = \{\{M_1, M_3, M_4\} \cup \{M_3, M_4, M_6\} \cup \{M_1, M_2, M_4\} \cup \{M_2, M_3, M_4\}\}$$
$$= \{M_1, M_2, M_3, M_4, M_6\}$$

此时，族群 1 需要的机器类型数可以确定为 $NM_1 = 5$。

对于族群 2 和 3，重复以上步骤，可以得到如下的结果：

$$SM_2 = \{M_2, M_3, M_4, M_5, M_6\}, \quad NM_2 = 5$$
$$SM_3 = \{M_1, M_2, M_3, M_4\}, \quad NM_3 = 4$$

染色体 1 的适应值可以计算求得为：

$$FF_1 = 5 + 5 + 4 = 14$$

以相似的方式，可以求得染色体 2,3,4 的适应值分别为 16,15 和 15。确定适应值后，下一步是按如下的步骤计算每一染色体的复制概率：

$$TFF = 14 + 16 + 15 + 15 = 60$$

$$AF_1 = 60/14 = 4.28, \quad AF_2 = 3.75, \quad AF_3 = 4.0, \quad AF_4 = 4.0$$

$$P_1 = 4.28/16.03 = 0.267, \quad P_2 = 0.234, \quad P_3 = 0.249, \quad P_4 = 0.249$$

其他结果 对前一节介绍的例子进行了求解。种群大小为 25，迭代世数为 500，求得的最好适应值为 10。计算结果如表 9.13 所示，由于三个解都对应于同一族群和单元，它们提供相同的结果。其中，频率代表运行中最优解求得的次数。

表 9.13 三个族群问题的各个解方案

解方案	适应值	频率	染色体	族群	单元中的机器
1	10	1	3223323331	(10)	[2,3]
				(6,3,2)	[3,4,5,6]
				(9,8,7,5,4,1)	[1,2,3,4]
2	10	1	1331131112	(9,8,7,5,4,1)	[1,2,3,4]
				(10)	[2,3]
				(6,3,2)	[3,4,5,6]
3	10	1	3113313332	(6,3,2)	[3,4,5,6]
				(10)	[2,3]
				(9,8,7,5,4,1)	[1,2,3,4]

9.6.2 族群数的确定

在有些场合，可能难以事先确定族群数，此时，设计者将不得不在变化的 f 值下运行遗传算法。

不同的族群数：本节将讨论 Süer 等人的结果[595]。在他们的分析计算中，对同一问题计算了 $f=2$ 和 $f=4$ 两种情况，计算结果分别列在表 9.14 和表 9.15。

表 9.14　2-族群解的结果

解方案	适应值	频率	染色体	族　群	单元中的机器
1	8	15	1221121111	(10,9,8,7,5,4,1)	[1,2,3,4]
				(6,3,2)	[3,4,5,6]
2	8	9	2112212222	(6,3,2)	[3,4,5,6]
				(10,9,8,7,6,5,4,3,2,1)	[1,2,3,4]
3	8	7	2222222221	(10)	[2,3]
				(9,8,7,6,5,4,3,2,1)	[1,2,3,4,5,6]
4	8	22	1111111112	(9,8,7,6,5,4,3,2,1)	[1,2,3,4,5,6]
				(10)	[2,3]

然而，没有足够的信息确定 2-族群解、3-族群解和 4-族群解中哪个更好，因此，在每个族群情况下有一些可选的解方案。即使，每个族群情况下的可选解方案得到相同的适应值，也难以确定哪个解方案较好。因而，必须计算机器数来比较各可选解方案，在计算中，处理时间和需求信息也是必要的。

表 9.15　4-族群解的结果

解方案	适应值	频率	染色体	族　群	单元中的机器
1	13	1	433434142	(8)	[1,2,3]
				(10)	[2,3]
				(6,3,2)	[3,4,5,6]
				(9,7,5,4,1)	[1,2,3,4]
2	13	1	3113313234	(6,3,2)	[3,4,5,6]
				(8)	[2,3]
				(9,7,5,4,1)	[1,2,3,4]
				(10)	[2,3]

一般方法　确定最优族群数和最好解方案的一般方法可以总结如下：

1. 令 $f=\text{MIN}$，可接受的最小族群数。
2. 应用进化规划将 n 个零部件指派给 f 个族群，使得机器类型的总数最小。

3. 确定机器数 M_f。

4. f 值增加一个单位，即，$f=f+1$。

5. 如果 $f \leqslant \mathrm{MAX}$（最大允许的族群数），则返回第 2 步。否则，进入第 6 步。

6. 可选解方案(g)是最后结构，其中，$M_g=\min\{M_f\}$，$\mathrm{MIN} \leqslant f \leqslant \mathrm{MAX}$。

机器数-基于机器瓶颈的方法 本节讨论 Süer 等人的基于机器瓶颈的方法[595]。该方法当使用机器间的单位传递时可行。由于物料处理等因素，没有零部件加工的机器可以闲置起来。使用基于瓶颈计算的另一个原因是所有时间内一个单元只有单个零部件的限制。显然，由于增加了机器闲置时间，这种方法需要较多的机器。假定每一单元内只需要每一种机器类型，零部件的生产速率按操作的最大加工时间计算，即

$$t_{i,\max} = \max_{j \in O_i}\{t_{ij}\} \tag{9.38}$$

其中，t_{ij} 是产品 i 在第 j 个加工操作上的处理时间，O_i 是产品 i 的加工操作集，$t_{i,\max}$ 是产品 i 的最大加工时间。满足一个族群中所有产品需要的总时间可以由产品需求和生产速率来确定：

$$TR_k = \sum_{i \in SP_k} \frac{d_i}{t_{i,\max}} \tag{9.39}$$

其中，d_i 是产品 i 年需求量，TR_k 是族群 k 需要的总时间。

族群 k 需要的单元数由总时间除以每一单元在一年内的可利用时间(A)，即：

$$NC_k = TR_k \,/\, A \tag{9.40}$$

其次，每一族群需要的机器数(MN_k)由单元数乘以一个单元内的机器类型数，即：

$$MN_k = NC_k \times MN_k \tag{9.41}$$

最后，系统需要的机器总数为所有族群需要的机器数之和：

$$M_f = \sum_{i=1}^{f} MN_i \tag{9.42}$$

数值例子 表 9.16 给出的数值例子是在前一节的数值例子上增加了处理时间和需求信息。产品 1 在机器 4 上的最大单位处理时间是 0.2h，因此，产品 1 从其单元的最大输出速率受限于机器 4，其速率为 5 每小时单位。如果产品 1 的年需求为 7000 个单位，即产品 1 需要 1400h 的单元。每个产品需要的小时数的计算结果包含在表 9.16 中。

表 9.16 处理时间和产品需求

机器	产品									
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
M_1	0.15			0.25	0.15			0.25		
M_2				0.20	0.10		0.15	0.20	0.30	0.15
M_3	0.18		0.25	0.30		0.25	0.20	0.14	0.25	0.20

续表

机　器	产　品									
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
M_4	0.20	0.10	0.20	0.28	0.11	0.20	0.15		0.20	
M_5		0.20				0.30				
M_6		0.12	0.15			0.15				
年需求	7k	4k	10k	6k	8k	6k	2.5k	2k	3k	2.5k
生产速率/h	5	5	4	3.33	6.66	3.33	5	4	3.33	5
需要时间/h	1400	800	2500	1800	1200	1800	500	500	900	500

求得的2-族群、3-族群和4-族群解结果分别列于表9.17、9.18和9.19之中。从表中可知，对于3-族群来说，所有可选方案的机器数都是30，对于4-族群来说，所有可选方案都是29。然而，对于2-族群来说，结果变化较大，有两个方案需要的机器数为28，有两个方案需要的机器数为38。在许多场合，需要一些相等的单元来满足这个需要。

表9.17　2适应值为8的2-族群解的结果

解方案	机器数	染色体	族　群	单元中的机器	单元
1	28	1221121111	(10,9,8,7,5,4,1)	[1,2,3,4]	4
			(6,3,2)	[3,4,5,6]	3
2	28	2112212222	(6,3,2)	[3,4,5,6]	3
			(10,9,8,7,5,4,1)	[1,2,3,4]	4
3	38	2222222221	(10)	[2,3]	1
			(9,8,7,6,5,4,3,2,1)	[1,2,3,4,5,6]	6
4	38	1111111112	(9,8,7,6,5,4,3,2,1)	[1,2,3,4,5,6]	6
			(10)	[2,3]	1

表9.18　适应值为10的3-族群解的结果

解方案	机器数	染色体	族　群	单元中的机器	单元
1	30	3223323331	(10)	[3,4]	1
			(6,3,2)	[3,4,5,6]	3
			(9,8,7,5,4,1)	[1,2,3,4]	4
2	30	1331131112	(9,8,7,5,4,1)	[1,2,3,4]	4
			(10)	[3,4]	1
			(6,3,2)	[3,4,5,6]	3
3	30	3113313332	(6,3,2)	[3,4,5,6]	3
			(10)	[2,3]	1
			(9,8,7,5,4,1)	[1,2,3,4]	4

表 9.19 适应值为 13 的 4-族群解的结果

解方案	机器数	染色体	族 群	单元中的机器	单元
1	29	4334434142	(8)	[1,2,3]	1
			(10)	[2,3]	1
			(6,3,2)	[3,4,5,6]	3
			(9, 7, 5,4, 1)	[1,2,3,4]	3
2	29	3113313234	(6,3,2)	[3,4,5,6]	3
			(8)	[1,2,3]	1
			(9, 7, 5,4, 1)	[1,2,3,4]	3
			(10)	[2,3]	1

表 9.17 中解方案 1 第 1 族群的零部件是 $P_1,P_4,P_5,P_7,P_8,P_9,P_{10}$。第 1 族群所有零部件需要的时间之和为

$$TR_1 = 1400 + 1800 + 1200 + 500 + 500 + 900 + 500 = 6800\text{h}$$

假定制造单元每年的可利用时间为 2000 小时，很容易确定需要 4 个单元来满足这个需求，即

$$NC_1 = \left\lceil \frac{6800}{2000} \right\rceil = 4 \text{ 单元}$$

其中，$\lceil x \rceil$ 表示比 x 大的最小整数。因此，需要的机器总数为单元数乘以单元内的机器数，即：

$$MN_1 = NC_1 \times NM_1 = 4 \times 4 = 16 \text{ 台机器}$$

类似地，TR_2、NC_2 和 MN_2 的计算值分别是 5100h，3 个单元和 12 台机器。因而，2-族群情况下需要的机器总数为

$$M_2 = 16 + 12 = 28 \text{ 台机器}$$

最好解方案：由于这种结构产生最小的机器数，因此，最好的解是 2-族群的零部件成组方案，即

$$M_g = \min\{28,30,29\} = 28 \text{ 台机器}, \qquad \text{MIN} \leqslant f \leqslant \text{MAX}$$

9.6.3 极小化机器数

Süer 等人[596]还直接使用适应值函数所要求的机器总数。所有其他运算都与前面的相同，并运算 10 次，机器计算的次数保持相同。同一问题遗传算法的种群大小为 25，迭代世代数为 500，求得如下的最好结果。在许多场合需要的机器总数低于 26，表 9.20 到表 9.22 给出了这些结果。这些结果还表明 25-机器、25-单元＋机器、10-单元＋15-机器、15-机器＋10-单元，以及 20-机器＋5-单元这几种策略产生完全相同的结果，并且优于5-机器＋20-单元和 25-单元的策略。

表 9.20 25-机器、25-单元+机器、10-单元+15-机器、15-机器+10-单元、20-机器+5-单元策略的解

结 果	族 群 数				
	2	3	4	5	6
最小值	27	26	26	26	26
最大值	27	26	26	26	26
平均值	27	26	26	26	26
最好解	0	10	10	10	10

表 9.21 25-单元策略的解

结 果	族 群 数				
	2	3	4	5	6
最小值	27	26	27	26	27
最大值	32	32	29	29	30
平均值	30	28.5	28.2	27.4	28.2
最好解	0	1	0	1	0

表 9.22 5-机器+20-单元策略的解

结 果	族 群 数				
	2	3	4	5	6
最小值	27	26	27	26	26
最大值	27	27	26	27	26
平均值	27	26.4	26	26.2	26
最好解	0	6	10	8	10

9.6.4 其他设想

在设计独立制造单元时，其他如机器重量和机器成本等因素也能够包含在适应值函数中。显然，基于瓶颈数的机器计算并不是计算所需要机器数的惟一方法。另一种计算机器需求的可能方法是基于所需机器的各加工处理时间，它正好与基于机器瓶颈的思路相对。毫无疑问，这将需要相同或更少的机器数。最后，也可以重点在于设计余下的制造单元，根据定义这些制造单元期望能对付任何零部件和任何加工处理，因此，余下的族群就如其他族群一样对待，除了每一机器类型中至少有一个放置在余下的单元中。Süer 等人在早期的一篇文章中简要地讨论该问题[393]。

参考文献

[1] Abido, M. A., Intelligent techniques approach to power system identification and control, Ph. D. dissertation, University of Petroleum and Minerals (Saudi Arabia), 1997.

[2] Abuali, F. N., Using determinant and cycle basis schemes in genetic algorithms for graph and network applications, Ph. D. dissertation, University of Tulsa, 1995.

[3] Abuali, F., R. Wainwright, and D. Schoenefeld, A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem, in Eshelman[177], pp. 470-475.

[4] Adams, J., E. Balas, and D. Zawack, The shifting bottleneck procedure for job shop scheduling, *International Journal of Flexible Manufacturing Systems*, vol. 34, no. 3, pp. 391-401, 1998.

[5] Adar, N., Allocation and scheduling on multi-computers using genetic algorithms, Ph. D. dissertation, Lehigh University, 1994.

[6] Adil, G. K., D. Ragamani, and D. Strong, Cell design considering alternative routings, *Int. Journal of Production Research*, vol. 34, no. 5, pp. 1361-1380, 1996.

[7] Aggarwal, K. K., Y. C. Chopra, and J. S. Bajwa, Topological layout of lips for optimizing the overall reliability in a computer communication system, *Microelectronics and Reliability*, vol. 22, no. 3, pp. 347-351, 1982.

[8] Aggarwal, K. K. and S. Rai, Reliability evaluation of computer-communication networks, *IEEE Transactions on Reliability*, vol. 30, no. 1, pp. 32-35, 1981.

[9] Ahuja, R. K., T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, NJ, 1993.

[10] Al-harkan, I. M., Onmerging sequencing and scheduling theory with genetic algorithms to solve stochastic job shop problems, Ph. D. dissertation, University of Oklahoma, 1997.

[11] Alander, J., *An Indexed Bibliography of Genetic Algorithms: 1957-1993*, Art of CAD Ltd., Espoo, Finland, 1994.

[12] Alvarez-Valdés, R. and J. Tamarit, Heuristic algorithms for resource constrained project scheduling: a review and an empirical analysis, in Slowinski, R. and J. Weglarz, editors, *Advances in Project Scheduling*, pp. 113-134, Elsevier Science Publishers, Amsterdam, 1989.

[13] Anderson, C., K. Jones, and J. Ryan, A two-dimensional genetic algorithm for the ising problem, *Complex Systems*, vol. 5, pp. 327-333, 1991.

[14] Aneja, Y. and K. Nair, Bicriteria transportation problem, *Management Science*, vol. 25, pp. 73-78, 1978.

[15] Ann, B. H. and J. -H. Hyun, Single facility multi-class job scheduling, *Computers and Operations Research*, vol. 17, pp. 265-272, 1989.

[16] Annaiyappa, P. V., Critical analysis of genetic algorithms for global optimization, Ph. D. dissertation, New Mexico State University 1991.

[17] Applegate, D. and W. Cook, A computational study of the job shop scheduling problem, *ORSA*

Journal of Computing, vol. 3, no. 2, pp. 149 156, 1991.

[18] Arabeyre, T., J. Faernley, F. Steiger, and W. Teather, The airline crew scheduling problem, *Transportation Science*, vol. 3, no. 2, pp. 140-163, 1969.

[19] Areibi, S., Toward optimal circuit layout using advanced search techniques, Ph. D. dissertation, University of Waterloo(Canada), 1995.

[20] Arguelles, D., Hybrid artificial neutral network/genetic algorithm approach to the on-line optimization of electrical power systems, Ph. D. dissertation, George Washington University, 1996.

[21] Askin, R. and K. Chiu, A graph partitioning procedure for machine assignment and cell formation, *Int. journal of Production Research*, vol. 28, no. 8, pp. 1555-1572, 1990.

[22] Atiqullah, M. M. and S. S. Rao, Reliability optimization of communication networks using simulated annaeling, *Microelectronics and Reliabiliy*, vol. 33, no. 9, pp. 1303-1319, 1993.

[23] Awadh, B., N. Sepehri, and O. Hawaleshka, A computer aided process planning model based on genetic alogorithms, *Computers and Operations Research*, vol. 22, no. 8, pp. 841-856, 1995.

[24] Awadth, B., Manufacturing process planning model using genetic algorithms, Ph. D. dissertation, University of Manitoba, 1994.

[25] Bäck, T., GENEsYs 1. 0 software distribution and installation notes, Technical report, Systems Analysis Research Group, LSXI, Department of Computer Science, University of Dortmund, Germany, 1992.

[26] Bäck, T., Optimal mutation rates in genetic search, in Forrest[204], pp. 2-9.

[27] Bäck, T., Selective pressure in evolutionary algorithms: a characterization of selection mechanisms, in Fogel[197], pp. 57-62.

[28] Bäck, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, 1996.

[29] Bäck, T., editor, *Proceedings of the 7th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Francisco, 1997.

[30] Bäck, T. and F. Hoffmeister, Extended selection mechanisms in genetic algorithms, in Belew and Booker[51], pp. 92-99.

[31] Bäcker, T., F. Hoffmeister, and H. Schwefel, A survey of evolution strategy, in Belew and Booker [51], pp. 2-9.

[32] Back, T. and H. Schwefel, An overview of evolution algorithms for parameter optimizations, *Evolutionary Computation*, vol. 1, no. 1, pp. 1-23, 1993.

[33] Bäck, T. and H. Schwefel, Evolutionary computation: a survey, in Fogel[195], pp. 20-28.

[34] Bakalis, O. L., Encoding the invariant measure of iterated affine transforms with a genetic algorithm, Ph. D. dissertation, University of New Mexico, 1996.

[35] Baker, J., Reducing bias and inefficiency in the selection algorithm, in Grefenstette[266], pp. 14-21.

[36] Baker, J., Adaptive selection methods for genetic algorithms, in Grefenstette [267], pp. 100-111.

[37] Baker, K., *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.

[38] Balas, E. and A. Ho, Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study, *Mathematical Programming Studies*, vol. 20, pp. 37-60, 1980.

[39] Balinski, M., Integer programming: methods, uses and computation, *Management Science*, vol. 12, no. 3, pp. 253-313, 1965.

[40] Balinski, M. and R. Quandt, On an integer program for a delivery problem, *Operations Research*, vol. 12, no. 2, pp. 300-304, 1964.

[41] Ball, M. and R. M. Van Slyke, Backtracking algorithms for network reliability analysis, *Annals of Discrete Mathematics*, vol. 1, pp. 49-64, 1977.

[42] Bangalore, S. S., Data analysis stragegies for qualitative and quantitative determination of organic compounds by Fourier transform infrared spectroscopy, Ph. D. dissertation, Ohio State University, 1996.

[43] Barr, R., R. Glover, and D. Klingman, A new optimization method for large scale fixed charge transportation problems, *Operations Research*, vol. 29, no. 3, pp. 448-463, 1981.

[44] Bazaraa, M., J. Jarvis, and H. Sherali, *Linear Programming and Network Flows*, 2nd edition, Wiley, New York, 1990.

[45] Bazaraa, M., H. Sherali, and C. Shetty, *Nonlinaer Programming: Theory and Algorithms*, 2nd edition, Wiley, New York, 1993.

[46] Bean, J. C., A. Lagrangian algorithm for the multiple choice integer program, *Operations Research*, vol. 32, pp. 1185-1193, 1984.

[47] Bean, J., Genetic algorithms and random keys for sequencing and optimization, *ORSA Journal of Computing*, vol. 6, no. 2, pp. 154-160, 1994.

[48] Beasley, J. E., OR-library: distributing test problems by electronic mail, *Journal of the Operations Research Society*, vol. 41, no. 11, pp. 1069-1072, 1990.

[49] Beasley, J. E. and P. C. Chu, A genetic algorithm for the set covering problem, *Europaen Journal of Operational Research*, vol. 94, pp. 392-404, 1996.

[50] Beasley, J. E. and K. Jornsten, Enhancing an algorithm for set covering problems, *Europaen Journal of Operational Research*, vol. 58, pp. 293-300, 1992.

[51] Belew, R. and L. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, Morgan Kaufman Publishers, San Francisco, 1991.

[52] Bellman, R. and L. Zadeh, Decision making in a fuzzy environment, *Management Science*, vol. 17, pp. 141-146, 1970.

[53] Bellmore, M., Set covering and involutional bases, *Management Science*, vol. 18, no. 3, pp. 194-206, 1971.

[54] Bellmore, M., H. Greenberg, and J. Jarvis, Multi-commodity networks, *Management Science*, vol. 16, no. 6, pp. 427-433, 1970.

[55] Berman, O., R. C. Larson, and S. S. Chiu, Optimal server location on a network operating as an $M/G/1$ queue, *Operations Research*, vol. 33, no. 4, pp. 746-771, 1985.

[56] Berman, O. and B. LeBlance, Location-relocation of N mobile facilities on a stochastic network, *Transportation Science*, vol. 21, pp. 207-216, 1984.

[57] Berman, O. and R. R. Mandowsky, Location-allocation on congested network, *European Journal of Operational Research*, vo. 26, pp. 238-250, 1986.

[58] Bertsekas, D. and P. Tseng, Relaxation methods for minimum cost oridinary and generalized network flow problems, *Operations Research*, vo. 36, no. 1, pp. 93-114, 1988.

[59] Bertismas, D., The probabilistic minimum spanning tree problem, *Networks*, vol. 20, pp. 245-275, 1990.

[60] Besson, N. W., Evaluation of the genetic algorithm as a computational tool in protein NMR, Ph. D. dissertation, Harvard University, 1995.

[61] Bhat, M. V. and A. Haupt, An efficient clustering algorithm, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no. 1, pp. 61-64, 1976.

[62] Billo, R., D. Tate, and B. Bidanda, A genetic cluster algorithm for the machinecomponent grouping problem, Technical report, University of Pittsburgh, 1995.

[63] Billo, R. E., D. Tate, and B. Bidanda, Comparison of a genetic algorithm and clustyer analysis for the cell formation problem: a case study, in *Proceedings of the 3rd Industrial Engineering Research Conference*, pp. 543-548, Atlanta, GA, 1994.

[64] Bjorklund, M. and A. Elldin, A practical method of calculation for certain types of complex common control systs, *Ericsson Technics*, vol. 20, pp. 3-75, 1964.

[65] Bjorndal, M., A Caprara, P. Cowling, F. Croce, H. Lourence, F. Malucelli, A. Orman, D. Pisinger, C. Rego, and J. Salazar, Some thoughts on combinatorial optimization, *European Journal of Operational Research*, vol. 83, pp. 253-270, 1995.

[66] Black, U., *TCP/IP and Related Protocols*, McGraw-Hill, New York, 1995.

[67] Blackstone, J., D. Phillips, and G. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations, *Int. Journal of Production Research*, vol. 20, pp. 27-45, 1982.

[68] Blom, G., *Probability and Statistics: Theory and Applications*, Discrete Applied Mathematics, 1996.

[69] Boctor, F., A linear formulation of the machine-part cell formation problem, *Int. Journal of Production Research*, vol. 29, no. 2, pp. 343-356, 1991.

[70] Bollobas, B., *Graph Theory: An Introductory Course*, Springer-Verlag, New York, 1979.

[71] Booker, L., Improving search in genetic algorithms, in Davis, L., editor, *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, San Francisco, 1987.

[72] Bowden, R. O., Genetic algorithm-based machine learning applied to the dynamic routing of discrete parts, Ph. D. dissertation, Mississippi State University, 1992.

[73] Boxman, O. J., J. W. Cohen, and N. Huffels, Approximations of mean waiting time in an $M/G/s$ queuing systems, *Operations Research*, vol. 27, pp. 1115-1127, 1979.

[74] Brady, R., Optimization strategies gleaned from biological evolution, *Nature*, vol. 317, no. 31, pp. 804-806, 1985.

[75] Brandeau, M. and S. Chiu, An overview of representative problems in location research, *Management Science*, vol. 35, pp. 645-673, 1989.

[76] Brindle, A., Genetic algorithms for function optimization. Ph. D. dissertation, University of Alberta-Edmondton, 1981.

[77] Brown, E. C., Using the facility location problem to explore operator policies and constraint-handling methods for genetic algorithms, Ph. D. dissertation, University of Virginia, 1996.

[78] Brown, W. G., Optimal rate concept acquisition using version spaces and genetic algorithms, Ph. D. dissertations, Wayne State University, 1994.

[79] Bui, T. N. and B. R. Moon, On multi-dimensional encoding/crossover, in Eshelman[177], pp. 49-56.

[80] Burbidge, J. L., An introduction of group technology, in *Seminar on Group Technology*, Turin, Italy, 1969.

[81] Burbidge, J. L., *The Introduction of Group Technology*, Halsted Press, division of Wiley, New York, 1975.

[82] Busacker, R. and T. Saaty, *Finite Graphs and Networks*, Martinus Nijhoff, New York, 1965.

[83] Cancela, H. and M. E. Khadiri, A recursive variance-reduction algorithm for estimating communication-network reliability, *IEEE Transactions on Reliability*, vol. 44, no. 4, pp. 595-602, 1995.

[84] Canpolate, N., Optimization of seasonal irrigation scheduling by genetic algorithms, Ph. D. dissertation, Oregon State University, 1997.

[85] Carse, B., T. C. Fogarty, and A. Munro, Adaptive distributed routing using evolutionary fuzzy control, in Belew and Booker[51], pp. 389-396.

[86] Caskey, K. R., Genetic algorithms and neutral networks applied to manufacturing scheduling, Ph. D. dissertation, University of Washington, 1993.

[87] Cavaretta, M. J., Cultural algorithms and real-valued function optimization, Ph. D. dissertation, Wayne State University, 1995.

[88] Cedeño, W., the multiniche crowding genetic algorithm: analysis and applications, Ph. D. dissertation, University of California-Davis, 1995.

[89] Ceneño, W. and V. R. Vemuri, Database design with genetic algorithms, in Dasgupta, D. and Z. Michalewicz, editors, *Multiple Criteria Decision Making: Theory and Applications*, pp. 189-206, Springer-Verlag, Heidelberg, 1997.

[90] Chaer, W. S., Mixture-of-experts approach to adaptive estimation. Ph. D. dissertation, University of Texas-Austin, 1996.

[91] Chan, H. M. and D. A. Milner, Direct clustering algorithm for group formation in cellular manufacture, *OMEGA: International Journal of Management Science*, vol. 1, no. 1, pp. 65-74, 1982.

[92] Chanas, S., Fuzzy programming in multiobjective linear programming: a parametric approach, *Fuzzy Sets and Systems*, vol. 29, pp. 303-313, 1989.

[93] Chandrasekharan, M. P. and R. Rajagopalan, An ideal seed non-hierarchical clustering

algorithm for cellular manufacturing, *Int. Journal of Production Research*, vol. 24, no. 2, pp. 451-464, 1986.

[94] Chandy, K. M. and T. Lo, The capacitated minimum spanning tree, *Networks*, vol. 3, pp. 173-182, 1973.

[95] Chang, T. C. and R. A. Wysk, *An Introduction to Automated Process Planning Systems*, Prentice Hall, Upper Saddle River, NJ, 1985.

[96] Chankong, A. and M. Miller, A model for optimal programming of railway freight train movements, *Management Science*, vol. 3, no. 1, pp. 74-92, 1956.

[97] Changkong, V. and Y. Y. Haimes, *Multiobjective Decision Making Theory and Methodology*, North-Holland, New York, 1983.

[98] Charnes, A. and W. W. Cooper, *Wanagement Models and Industrial Applications of Linear Programming*, Wiley, New York, 1961.

[99] Chbat, N. W., Direct-learning neutral and fuzzy control of two unknown dynamic systems using Powell optimization and genetic algorithms, Ph. D. dissertation, Columbia University, 1996.

[100] Cheng, C., Study on optimal design of fuzzy facility layout, master's thesis, Ashikaga Institute of Technology, March 1995.

[101] Cheng, C. H., Y. P. Gupta, W. H. Lee, and K. F. Wong, A TSP-based heuristic for forming machine groups and part families, *Int. Journal of Production Research*, vol. 28, no. 5, pp. 1325-1337, 1998.

[102] Cheng, M. Y., Control design and robustness measurement for biped locomotion, Ph. D. dissertation, University of Missouri-Columbia, 1996.

[103] Cheng, R., Study on genetic algorithm-based optimal scheduling techniques, Ph. D. dissertation, Tokyo Institute of Technology, 1997.

[104] Cheng, R. and M. Gen, A hybrid genetic algorithm for the parallel machine minmax weighted absolute lateness problems, *Engineering Desin and Automation* (forthcoming).

[105] Cheng, R. and M. Gen, Evolution program for resource constrained project scheduling problem, in Fogel[197], pp. 736-741.

[106] Cheng, R. and M. Gen, Resource constrained project scheduling problem using genetic algorithms, *International Journal of Intelligent Automation and Soft Computing*, vol. 3, no. 3, pp. 273-286, 1997.

[107] Cheng, R. and M. Gen, An adaptive superplane approach for multiple objective optimization problems, Technical report, Ashikaga Institute of Technology, 1998.

[108] Cheng, R. and M. Gen, Loop layout design problem in flexible manufacturing system using genetic algorithm, *Computers and Industrial Engineering*, vol. 34, no. 1, pp. 53-61, 1998.

[109] Cheng, R. and M. Gen, Compromise approach-based genetic algorithms for bicriterion shortest path problems, Technical report, Ashikaga Institute of Technology, 1998.

[110] Cheng, R. and M. Gen, An evolution program for the resource constrained project scheduling problem, *Computer Integrated Manufacturing*, vol. 11, no. 3, pp. 274-287, 1998.

[111] Cheng, R. and M. Gen, A priority based encoding and shortest path problem, Technical report,

Ashikaga Institute of Technology, 1998.

[112] Cheng, R. and M. Gen, A survey of genetic multiobjective optimizations, Technical report, Ashikaga Institute of Technology, 1998.

[113] Cheng, R., M. Gen, and T. Tozawa, Genetic algorithms for multi-row machine layout problem, *Engineering Design and Automation* (forthcoming).

[114] Cheng, R., M. Gen, and Y. Tsujimura, A tutorial survey of job-shop scheduling problems using genetic algorithms: Ⅰ. Representation, *Computers and Industrial Engineering*, vol. 30, pp. 983-997, 1996.

[115] Cheng, R., M. Gen, and Y. Tsujimura, A tutorial survey of job-shop scheduling problems using genetic algorithms: Ⅱ. Hybrid genetic search strategies, *Computers and Industrial Engineering*, vol. 36, no. 2, 1999.

[116] Cheng, S. J. and C. S. Cheng, A neural network-based cell formation algorithm in cellular manufacturing, *Int. Journal of Production Research*, vol. 33, no. 2, pp. 293-318. 1995.

[117] Cheng, T. and X. Cai, On the complexity of completion time variance minimization problem. Working paper. University of Western Australia-Nedlands, 1990.

[118] Chintrakulchai, P., Hign-performance fractal image compression, Ph. D. dissertation, Dartmouth College, 1995.

[119] Chiu, S. S., O. Berman, and R. C. Larson, Locating a mobile server queueing facility on a tree network, *Management science*, vol. 31, no. 6, pp. 764-772, 1985.

[120] Chopra, Y. C., B. S. Sohi, R. K. Tiwari, and K. K. Aggarwal, Network topology for maximizing the terminal reliability in a computer communication network, *Microelectronics and Reliability*, vol. 24, pp. 911-913, 1984.

[121] Christofides, N., R. Alvarez-Valdés, and J. Tamarit, Project scheduling with resource constraint: a branch bound approach, *European Journal of Operational Research*, vol. 29, pp. 262-273, 1987.

[122] Chu, T., Some problems in fuzzy decision making, Ph. D. dissertation, University of Texas-Arlington, 1993.

[123] Chunduru, R. K., Global and Hybrid optimization in geophysical inversion, Ph. D. dissertation, University of Texas-Austin. 1998.

[124] Chung, W. S., Analysis of the effects of different representation schemes on genetic algorithms, Ph. D. dissertations, University of South Florida, 1995.

[125] Chimaco, J. and E. Martins, A bicriterion shortest path algorithm, *European Journal of Operational Research*, vol. 11, pp. 399-404, 1982.

[126] Cobham, A., R. Fridshal, and J. North, An application of linear programming to the minimization of Boolean functions, Report RC 472, IBM Research Center, 1961.

[127] Cohoon, P. and W. Paris, Genetic placement, in *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 422-425, 1986.

[128] Coit, D. W. and A. E. Smith, Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach, *Computers and Operations Research*, vol. 23,

no. 6, pp. 515-526, 1996.

[129] Coit, D., Optimization of reliability design problems considering uncertainty in component reliability and time-to failure, Ph. D. dissertation, University of Pittsburgh, 1996.

[130] Coit, D. W. and A. E. Smith, Redundancy allocation to maximize a lower precentile of the system time-to-failure distribution, *IEEE Transactions on Reliability*, vol. 47, no. 1, pp. 79-87, 1998.

[131] Collins, R. J., Studies in artificial evolution, Ph. D. dissertation, University of California-Los Angeles, 1992.

[132] Comer, D. E., *Internetworking with TCP/IP*, vol. 1: *Principles, Protocols, and Architecture*, 3rd edition, Prentice Hall, Upper Saddle River, NJ, 1995.

[133] Corcoran, A. L., Techniques for reducing the disruption of superior building blocks in genetic algorithms, Ph. D. dissertation, University of Tulsa, 1994.

[134] Cormen, T. H., C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[135] Cowgill, M. C., Monte Carlo validation of two genetic clustering algorithms, Ph. D. dissertation, Virginia Polytechnic Institute and State University, 1993.

[136] Cox, L. A., Dynamic anticipatory routing in circuit-switched telecommunications networks, in Davis[147], pp. 124-143.

[137] Crawford, K. D., Role of recombination in genetic algorithms for fixed-length subset problems, Ph. D. dissertation, University of Tulsa, 1996.

[138] Crossley, W. A., Using genetic algorithms as an automated methodology forconceptual design of rotorcraft, Ph. D. dissertation, Arizona State University, 1995.

[139] Daskin, M. S., An overview of recent research on assigning products to groups for group technology production problems, Technical report, Northwestern University, 1991.

[140] Davern, J. J., Architecture for job shop scheduling with genetic algorithms, Ph. D. dissertation, University of Central Florida, 1994.

[141] Davidor, Y., A genetic algorithm applied to robot trajectory generation, in Davis[147], pp. 923-932.

[142] Davidor, Y., H. Schwefel, and R. Männer, editors, *Parallel Problem Solving fron Nature: PPSN Ⅲ*, Springer-Verlag, Berlin, 1994.

[143] Davis, E. and G. Heidorn, An algorithm for optimal project scheduling under multiple resources constraints, *Management Science*, vol. 17, pp. B803-816, 1971.

[144] Davis, L., Applying adaptive algorithms to epistatic domains, *in Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 162-164, 1985.

[145] Davis, L., Job shop scheduling with genetic algorithm, in Grefenstette[266], pp. 136-140.

[146] Davis, L., Adapting operator Probabilities in genetic algorithm, in Schaffer[564], pp. 61-69.

[147] Davis, L., editor, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[148] Davis, P. and T. Ray, A branch-bound algorithm for the capacitated facilities location problem,

Naval Research Logistics Quarterly, vol. 16, pp. 331 344, 1969.

[149] Dawkind, R., *The Selfish Gene*, Oxford University Press, Oxford. 1976.

[150] Day, R., On optimal extraction from a multiple file data storage system, *Operations Research*, vol. 13, no. 3, pp. 428-494, 1995.

[151] Deb, K., Genetic algorithms in multimodel function optimization, M. S. dissertation, University of Alabama, 1989.

[152] DeChaine, M. D., Stochastic fuel management optimization using genetic algorithms and heuristic rules, Ph. D. dissertation, Pennsylvania State University, 1995.

[153] Deeter, D. L. and A. E. Smith, Heuristic optimization of network design considering all-terminal reliability, in *Proceedings of the Annual Reliability and Maintainability Symposium*, pp. 194-199, 1997.

[154] Deeter, D. L, and A. E. Smith, Economic design of reliable network, *IIE Transactions*, vol. 30, pp. 1161-1174, 1998.

[155] Demmeyer, F. and S. Voss, Dynamic tabulist management using the reverse elimination method, *Annals of Operations Research*, vol. 41, pp. 31-46, 1993.

[156] Dengiz, B., F. Altiparmak, and A. E. Smith, A genetic algorithm approach to optimal topology design of all terminal networks, in *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 5, pp. 405-410, ASME Press, New York, 1995.

[157] Dengiz, B., F. Altiparmak, and A. E. Smith, Efficient optimization of allterminal reliable networks using evolutionary approach, *IEEE Transactions on Reliability*, vol. 46, no. 1, pp. 18 26, 1997.

[158] Dengiz, B., F. Altiparmak, and A. E. Smith, Local search genetic algorithm for optimal design of reliable networks, *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 3, pp. 179-188, 1997.

[159] Dengiz, B., F. Altiparmak, and A. E. Smith, Local search gengtic algorithm for optimization of highly reliable communications networks, in Bäck[29], pp. 650-657.

[160] Deo, N. and N. Kumar, Computation of constrained spanning trees, in Pardalos, P. M., editor, *Network Optimization*, pp. 194-233, Springer-Verlag, Berlin, 1997.

[161] Dev, K., *Optimization for Engineering Design; Algorithms and Examples*, Prentice-Hall, New Delhi, 1995.

[162] Dhingra, A. K., Optimal apportionment of reliability and redundancy in a series system under multiple objectives, *IEEE Transactions on Reliability*, vol. 41, no. 4, pp. 576-582, 1992.

[163] Dighe, R., Human pattern nesting strategies in a genetic algorithms framework, Ph. D. dissertation, Massachusetts Institute of Technology, 1996.

[164] Dijkstra, E., A note on two problems in connection with graphs, *Numerische Mathematick*, vol. 1, pp. 269-271, 1959.

[165] Dorndorf, U. and E. Pesch, Evolution based learning in a job shop scheduling environment, *Computers and Operations Research*, vol. 22, no. 1, pp. 25-40, 1995.

[166] Dozier, G. V., Constraint processing using adaptive microevolutionary/systematic hill climbing,

Ph. D. dissertation, North Carolina State University, 1995.

[167] Drexl, A. and J. Gruenewald, Nonpreemptive multi-mode resource constrained project scheduling, *IIE Transactions*, vol. 25, pp. 74-81, 1993.

[168] Dudzinski, K. and S. Walukiewicz, Exact methods for the knapsack problem and its generalizations, *European Journal of Operational Research*, vol. 28, pp. 3-21, 1987.

[169] Eberlein, M. V., GA heuristic generically has hyperbolic fixed points, Ph. D. dissertation, University of Tennessee, 1996.

[170] Edirisinghe, C. D., Optimization of radiotherapy planning using genetic algorithms, Ph. D. dissertation, State University of New York-Buffalo, 1996.

[171] Efroymson, M. and T. Ray, A branch-bound algorithm for plant location, *Operations Research*, vol. 14, pp. 361-368, 1966.

[172] Egan, M. A., Validity-guided robust fuzzy clustering methods for characterizing clusters in noisy data, Ph. D. dissertation, Rensselaer Polytechnic Institute, 1997.

[173] Eichler, W. R. M., On the development and interpretation of parameter manifolds for biophysically robust compartmental models of CA3 hippocampal neurons, Ph. D. dissertation, University of Minnesota, 1996.

[174] Elbaum, R. and M. Sidi, Topological design of local-area networks using genetic algorithms, *IEEE/ACM Transactions on Networking*, vol. 4, no. 5, pp. 766-778, 1996.

[175] Elias, D. and M. Ferguson, Topological design of multipoint teleprocessing networks, *IEEE Transactions on Communications*, vol. 22, pp. 1753-1762, 1974.

[176] Elketroussi, M., Relapse from tobacco smoking cessation: mathematical and computer microsimulation modeling including parameter optimization with genetic algorithms, Ph. D. dissertation, University of Minnesota, 1993.

[177] Eshelman, L. J., editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, Morgan Kaufmannn Publishers, San Francisco, 1995.

[178] Eshelman, L., K. Mathias, and J. Schaffer, Convergence controlled variation, in Belew, R. and M. Vose, editors, *Foundations of Genetic Algorithms*, vol. 4, Morgan Kaufmann Publishers, San Francisco, 1997.

[179] Eshelman, L., K. Mathias, and J. Schaffer, Crossover operator biases: exploiting the population distributions, in Bäck[29], pp. 354-361.

[180] Eshelman, L. and J. Schaffer, Real-coded genetic algorithms and intervalschemata, in Whitley, L., editor, *Foundations of Genetic Algorithms*, vol. 2, pp. 187-202, Morgan Kaufmann Publishers, San Francisco, 1993.

[181] Falkenauer, E., A new representation and operators for genetic algorithms applied to grouping problems, *Evolutionary Computation*, vol. 2, no. 2, pp. 123-144, 1994.

[182] Falkenauer, E., Tapping the full power of genetic algorithms through suitable representation and local optimization: application to bin packing, in *Evolutionary Algorithms in Management Applications*, pp. 167-182, Springer-Verlag, Berlin, 1995.

[183] Falkenauer, E. and S. Bouffoix, A genetic algorithm for job shops, in *Proceedings of the IEEE*

International Conference on Robotics and Automation, pp. 824-829, 1991.

[184] Falkenauer, E. and A. Delchambre, A genetic algorithm for bin packing and line balancing, in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1186-1192, 1992.

[185] Fazakerley, G. M., A research report on the human aspects of group technology and cellular manufacture, *Int. Journal of Production Research*, vol. 14, no. 1, pp. 123-134, 1976.

[186] Fernandes, L. M. and L. Gouveia, Minimal spanning trees with a constraint on the number of leaves, *European Journal of Operational Research*, vol. 104, pp. 250-261, 1998.

[187] Fetterlof, P. C. and G. Anandalingam, Optimal design of LAN-WAN internetworks: an approach using simulated annealing, *Annals of Operations Research*, vol. 36, pp. 275-298, 1992.

[188] Filho, A. A., Optimizing hydrocarbon field development using a genetic algorithm-based approach, Ph. D. dissertation, Stanford University, 1997.

[189] Fisher, M. L. and P. Kedia, Optimal solution of set covering/partitioning problems using dual heuristics, *Management Science*, vol. 36, pp. 674-688, 1990.

[190] Fishman, G. S., A comparison of four Monte Carlo methods for estimating the probability of s-t connectedness, *IEEE Transactions on Reliability*, vol. R-35, no. 2, pp. 145-155, 1986.

[191] Fishman, G. S., A Monte Carlo sampling plan for estimating network reliability, *Annals of Operations Research*, vol. 34, pp. 581-594, 1986.

[192] Fluerent, C., *Algorithmes génétiques hybrides pour l'optimisation combinatoire*, Ph. D. dissertation, University of Saskatchewan, 1994.

[193] Forgarty, T., Varying the probability of mutation in the genetic algorithm, in Schaffer[564], pp. 104-109.

[194] Fogarty, T., editor, *Evolutionary Computing*, Springer-Verlag, Berlin, 1994.

[195] Fogel, D., Editor, *Proceedings of the IEEE International Conference on Evolutionary Computation*, IEEE Press, Piscataway, NJ, 1996.

[196] Fogel, D., editor, *Proceedings of the IEEE International Conference on Evolutionary Computation*, IEEE Press, Piscataway, NJ, 1998.

[197] Fogel, D., editor, *Proceedings of the First IEEE Conference on Evolutionary Computation*, IEEE Press, Piscataway, NJ, 1994.

[198] Fogel, D., *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, 1995.

[199] Fogel, D. and J. Atmar, Comparing genetic operators with Gaussian mutations in simulated evolutionary process using linear systems, *Biological Cybernetics*, vol. 63, pp. 111-114, 1990.

[200] Fogel, L., A. Owens, and M. Walsh, *Artificial Intelligence Through Simulated Evolution*, Wiley, New York, 1966.

[201] Fonseca, C. and P. Fleming, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, in Forrest[204], pp. 416-423.

[202] Fonseca, C. and P. Fleming, An overview of evolutionary algorithms in multiobjective

optimization, *Evolutionary Computation*, vol. 3, no. 1, pp. 1-16, 1995.

[203] Forrest, S., Documentation for prisoners dilemma and norms programs that use the genetic algorithm, Ph. D. dissertation, University of Michigan-Ann Arbor, 1985.

[204] Forrest, S., editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Francisco, 1993.

[205] Fourman, M., Compaction of symbolic layout using genetic algorithms, in Grefenstette[266], pp. 141-153.

[206] Freeman, D. W., Genetic algorithms: a new technique for solving the neutron spectrum unfolding problem, Ph. D. dissertation, University of Missouri-Rolla, 1998.

[207] Fujimoto, Y., Study on parallel processing architecture of neural networks and genetic algorithms, Ph. D. dissertation, Osaka University, 1993.

[208] Garey, M. and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, 1979.

[209] Garfinkel, R., Optimal political districting, Ph. D. dissertation, Johns Hopkins University, 1968.

[210] Gavish, B., Topological design or centralized computer networks: formulation and algorithms, *Networks*, vol. 12, pp. 355 377, 1982.

[211] Gavish, B., Formulation and algorithms for the capacitated minimal directed tree problem. *Journal of the Association for Computing Machinery*, vol. 30, no. 1, pp. 118-132, 1983.

[212] Gavish, B., Augmented Lagrangean based algorithms for centralized network design, *IEEE Transactions on Communications*, vol. 33, pp. 1247-1257, 1985.

[213] Gen, M. Reliability optimizaton by 0-1 programming for a system with several failure models, *IEEE Transactions on Reliability*, vol. 24, pp. 206-210, 1975; also in Rai, S. and D. P. Agrawal, editors, *Distributed Computing Network Reliability*, pp. 252-256, IEEE Computer Society Press, Los Alamitos, CA, 1990.

[214] Gen, M. and R. Cheng, Interval programming using genetic algorithms, in Gen, M. and Y. Tsujimura, editors, *Evolutionary Computations and Intelligent Systems*, Gordon and Breach, New York(forthcoming).

[215] Gen, M. and R. Cheng, Optimal design of system reliability under uncertainty using interval programming and genetic algorithm, Technical report, ISE94-6, Intelligent Systems Engineering Laboratory, Ashikaga Institute of Technology, 1994.

[216] Gen, M. and R. Cheng, Interval programming using genetic algorithms, in Jamshidi, M., M. Fathi, and F. Pierrot, editors, *Intelligent Automation and Control*, vol. 4, pp. 243 248, TSI Press, Albuquerque, NM, 1996.

[217] Gen, M. and R. Cheng, Optimal design of system reliability using interval programming and genetic algorithms, *Computers and Industrial Engineering*, vol. 31, no. 1-2, pp. 237-240, 1996.

[218] Gen, M. and R. Cheng, A survey of penalty techniques in genetic algorithms, in Fogel[195], pp. 804-809.

[219] Gen, M. and R. Cheng, *Genetic Algorithms and Engineering Design*, Wiley, New York. 1997.

[220] Gen, M., R. Cheng, M. Sasaki, and Y. Jin. Multiple-choice knapsack problem using genetic algorithms. In Parsaei, H., M. Gen, Y. Tsujimura, and J. Wong, editors, *Advances in Engineering Design and Automation Research II*, Integrated Technology Systems, Maui, HI, 1998.

[221] Gen. M., R. Cheng, and D. Wang. Genetic algorithms for solving shortest path problems, in Porto[512], pp. 401-406.

[222] Gen, M. and K. Ida. *Goal Programming Using Turbo C*, Harcourt Brace Japan, Tokyo, 1993 (in Japanese).

[223] Gen, M., K. Ida, and J. R. Kim, A spanning tree-based genetic algorithm for bicriteria topological network design, in Fogel[196], pp. 15-20.

[224] Gen, M., K. Ida, and J. R. Kim, System reliability optimization with fuzzy goals using genetic algorithm, *Journal of the Japan Society of Fuzzy Theory and Systems*, vol. 10, no. 2, pp. 356-365, 1998.

[225] Gen, M., K. Ida, J. R. Kim, and J. U. Lee, Bicriteria network desing using spanning tree-based genetic algorithm, in *Proceedings of the 3rd International Symposium on Artificial Life and Robotics*, vol. 1, pp. 43-46, 1998.

[226] Gen, M., K. Ida, and Y. Z. Li. Bicriteria transportation problem by hybrid genetic algorithm, *Computers and Industrial Engineering*, vol. 35, no. 1-2, pp. 363-366, 1998.

[227] Gen, M. and J. R. Kim, Bicriteria reliability design using hybrid genetic algorithm, in *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing*, vol. 1, pp. 407-412, 1998.

[228] Gen, M. and J. R. Kim, GA-based optimal network design, in Dagli, C. H., M. Akay, A. L. Buczak, and B. R. Fernandez, eds., *Intelligent Engineering Systems, Through Artificial Neural Networks*, vol. 8, pp. 247-252, ASME Press, New York, 1998.

[229] Gen, M. and J. R. Kim, GA-based optimisation of reliability design, in Bengley, P., *Evolutionary Design by Computers*, Chapter 8, pp. 191-218, Morgan Kaufmann, San Francisco, 1999.

[230] Gen, M. J. R. Kim, and Y. Li, Hybrid genetic algorithm for solving bicriteria reliability design problems, Technical report, Ashikaga Institute of Technology, 1998.

[231] Gen, M. and T. Kobayashi, editors, *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, Ashikaga, Japan, 1994.

[232] Gen, M. and Y. Li, Hybrid genetic algorithms for transportation problem in logistics, *Journal of Logistics* (forthcoming).

[233] Gen, M. and Y. Li, Solving multi objective transportation problem by spanning tree-based genetic algorithm, in Parmee, I., editor, *Adaptive Computing in Design and Manufacture*, pp. 95-108, Springer-Verlag, New York, 1998.

[234] Gen, M. and Y. Li, Spanning tree-based genetic algorithm for bicriteria fixed charge

transportation problem, in *Proceedings of the Congress on Evolutionary Computation*, pp. 2265-2271 Washington, DC, 1999.

[235] Gen, M. and B. Liu, Evolution algorithm for optimal capacity expansion, *Journal of the Operations Research Society of Japan*, vol. 40, no. 1, pp. 1-9,1997.

[236] Gen, M. and B. Liu, A genetic algorithm for nonlinear goal programming, *Evolutionary Optimization*, vol. 1, no. 1, 1999.

[237] Gen, M., B. Liu, and K. Ida, Evolution Program for deterministic and stochastic optimizations, *European Journal of Operational Research*, vol. 94, no. 3, pp. 618-625, 1996.

[238] Gen, M., B. Liu, K. Ida, and D. Zheng, Evolutin program for constrained nonlinear optimization, in Gen and Kobayashi[231], pp. 576-579.

[239] Gen, M. and Y. Tsujimura, editors. *Genetic Algorithms and Intelligent Systems*, Gordon and Breach, New York, (forthcoming).

[240] Gen, M., Y. Tsujimura, and E. Kubota, Solving job-shop scheduling problem using genetic algorithms, in Gen and Kobayashi[231], pp. 576-579.

[241] Gen, M., G. Zhou, and J. R. Kim, Genetic algorithma for solving network design problems: state-of-the-art survey, *Evolutionary Optimization*, vol. 1, no. 3, 1999.

[242] Gen, M., G, Zhou, and M. Takayama, Matrix based genetic algorithm approach on bicriteria minimum spanning tree problem with interval coefficients, *Journal of the Japan Society for Fuzzy Theory and Systems*, vol. 10, no. 6, pp. 1144-1153, 1998(in Japanese).

[243] Geoffrion, A, M., Lagrangian relaxation for integer programming, *Mathematical Programming*, PP. 82-114, 1974.

[244] Giddens, T. D., Determination of invariant parameters and operators of the traditional genetic algorithm using an adaptive genetic algorithm generator, Ph. D. dissertation, Texas Tech University, 1994.

[245] Gillies, A., Machine learning procedures for generating image domain feature detectors, Ph. D. dissertation, University of Michigan—Ann Arbor, 1985.

[246] Glover, F., M. Lee, and J. Ryan, Least-cost network topology design for a new service: an application of a tabu search, *Annuals of Operations Research*, vol. 33, pp. 351-362, 1991.

[247] Gold, F. M., Application of the genetic algorithm to the kinematic design of turbine blade fixtures, Ph. D. dissertation, Worcester Polytechnic Institute, 1998.

[248] Goldberg, D. and J. Richardson, Genetic algorithms with sharing for multimodal function optimization, in Grefenstette[267], pp. 41-49.

[249] Goldberg, D., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[250] Goldberg, D. and K. Deb, A comparative analysis of selection schemes used in genetic algorithms, in Rawlins[529], pp. 69-93.

[251] Goldberg, D., B. Korb, and K. Deb, Messy genetic algorithms: motivation, analysis, and first results, *Complex Systems*, vol. 3, pp. 493-530, 1989.

[252] Goldberg, D, and R. Lingle, Alleles, loci and the traveling salesman problem, in Grefenstette

[266], pp. 154-159.

[253] Golden, J. B., Adaptive approximation and optimization of transform functions, Ph. D. dissertation, Vanderbilt University, 1995.

[254] Gong, D., G. Yamazaki, M. Gen, and W. Xu, Hybrid evolutionary method for capacitated location-allocation problem, *Engineering Design and Automation*, vol. 3, no. 2, pp. 179 190, 1997.

[255] Gong, D., Evolutionary computation and artificual neural networks for optimization problems and their applications, Ph. D. dissertation. Tokyo Metropolitan Institute of Technology, 1998.

[256] Gong, D., G. Yamazaki, M. Gen, and W. Xu, A genetic method for onedimensional machine location problems, *International Journal of Production Economics*, vol. 60-61, pp. 337-342, 1999.

[257] González Hernández, L. F. and D. W. Corne, Evolutionary divide and conquer for the set-covering problem, in Fogarty, T., editor, *Evolutionary Computing*, pp. 198-208, Springer-Verlag, Berlin, 1996.

[258] Gordon, V. S., Exploitable parallelism in genetic algorithms, Ph. D. dissertation, Colorado State University, 1994.

[259] Gordon, V. and D. Whitley, Serial and parallel genetic algorithms as functions optimizers. in Forrest [204], pp. 177-183.

[260] Gottlieb, J. and L. Paulmann, Genetic algorithms for the fixed charge transportation problems, in *Proceedings of the IEEE International Conference Evolutionary Computation*, pp. 330-335, Anchorage, 1998.

[261] Gouveia, L., A $2n$ constraint formulation for the capacitated minimal spanning tree problem, *Operations Research*, vol. 43, no. 1, pp. 130-141, 1995.

[262] Graham, R. and P. Hell, On the history of the minimum spanning tree problem, *Annals of the History of Computing*, vol. 7, pp. 43-57, 1985.

[263] Grand, S. M., Application of the genetic algorithm to protein tertiary structure prediction, Ph. D. dissertation, Pennsylvania State University, 1993.

[264] Graybeal, M. E., Pest and resistance management simulation model of the Colorado potato beetle on potatoes, Ph. D. dissertation, Virginia Commonwealth University, 1998.

[265] Grefenstette, J. J., A user's guide to GENESIS, Technical report, Computer Science Department, Vanderbilt University, 1983.

[266] Grefenstette, J. J., editor, *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, N. J., 1985.

[267] Grefenstette, J. J., editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, N. J., 1987.

[268] Grefenstette, J. J., Optimization of control parameters for genetic algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, pp. 122-128, 1986.

[269] Grefenstette, J. J., Lamarkian learning in multi-agent environment, in Belew and Booker[51], pp. 303-310.

[270] Grefenstette, J. J. and J. Baker, How genetic algorithms work: a critical look at implicit parallelism, in Schaffer[564], pp. 20-27.

[271] Guan, J., Applications of genetic algorithms in groundwater quality management, Ph. D. dissertation, Georgia Institute of Technology, 1998.

[272] Guo, Z., Nuclear power plant fault diagnostics and thermal performance studies using neural networks and genetyic algorithms, Ph. D. dissertation, University of Tennessee, 1992.

[273] Gupta, T. and H. Seifoddini, Production data based similarity coefficient for machine-component grouping decisions in the design of a cellular manufacturing system, *Int. Journal of Production Research*, vol. 28, no. 7, pp. 1247-1269, 1990.

[274] Gupta, Y., M. Gupta, A. Kumar, and C. Sundram, Minimizing total intercell and intracell moves in cellular manufacturing: a genetic algorithm approach, *Computer Integrated Manufacturing*, Vol. 8, no. 2, pp. 92-101, 1995.

[275] Hachino, T., Study on identification of continuous time-delay systems, and lower dimensionizing of models by genetic algorithms, Ph. D. dissertation, Kyushu Institute of Technology, 1995(in Japanese).

[276] Hajda, P., Genetic algorithms for control of wastewater conveyance systems, Ph. D. dissertation, Marquette University, 1997.

[277] Hall, L., Experience with a cutting plane algorithm for the capacitated spanning tree problem, *INFORMS Journal on Computing*, vol. 8, no. 3, pp. 219-234, 1996.

[278] Ham, I. V. and C. Han, Multiobjective cluster analysis for part family formations, *OMEGA: International Journal of Management Science*, vol. 5, no. 4, pp. 223-229, 1986.

[279] Han, S. S., Modeling and optimization of plasma-enhanced chemical vapor deposition using neural networks and genetic algorithms, Ph. D. dissertation, Georgia Institute of Technology, 1996.

[280] Hanagandi, V., Process control, optimization, and modeling of chemical systems using genetic algorithms and neutral networks, Ph. D. dissertation, Texas A&M University, 1995.

[281] Hancock, P., An empirical comparison of selection methods in evolutionary algorithms, in Fogarty, T., editor, *Evolutionary Computing*, pp. 80-95, Springer-Verlag, Berlin, 1994.

[282] Hansen, P., Bicriterion path problems, in Fandel, G. and T. Gal, editors, *Multiple Criteria Decision Making: Theory and Applications*, pp. 109 127, Springer-Verlag, Heidelberg, 1980.

[283] Hanzhong, C. and L. Dongkui, A new algorithm for computing the reliability of complex networks bu the cut method, *Microelectronics and Reliability*, vol. 34, no. 1, pp. 175-177, 1994.

[284] Harche, F. and G. L. Thompson, The column subtraction algorithm: an exact method for solving weighted set covering, packing and partitioning problems, *Computers and Operations Research*, vol. 21, pp. 689-705, 1994.

[285] Harhalakis, G., R. Nagi, and J. M. Proth, An efficient heuristic in manufacturing cell formation for group technology applications, *Int. Journal of Production Research*, vol. 28, no. 1, pp. 185-198, 1990.

[286] Harik, G., Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms, Ph. D. dissertation, University of Michigan, 1997.

[287] Hart, W. E., Adaptive global optimization with local search, Ph. D. dissertation, University of California-San Diego, 1994.

[288] Hase, K., Simulation on mutual adaptation of a body form and biped walking by nerve muscle skelation model and genetic algorithms, Ph. D. dissertation, Keio University, 1996.

[289] Hashimoto, Y., Study of neural network and genetic algorithms on an application to production planning problems, Ph. D. dissertation, Ritsumeikan University, 1995.

[290] Haupt, R., A survey of priority-rule based scheduling problems, *OR Spectrum*, vol. 11, pp. 3-16, 1989.

[291] Healy, W. C., Multiple choice programming, operations research, *Computers and Operations Research*, vol. 12, pp. 122-138, 1964.

[292] Hedrick, C., *RFC-1058*; *Routing Information Protocol*, Network Working Group, New York, 1988.

[293] Henig, M., The shortest path problem with two objective functions, *European Journal of Operational Research*, vol. 25, pp. 281-291, 1986.

[294] Herrera, F. and M. Lozano, Adaptation of genetic algorithm parameters based on fuzzy logic controllers, in Herrera, F. and J. Verdegay, editors, *Genetic Algorithms and Soft Computing*, pp. 95-125, Physica-Verlag, 1996.

[295] Herrera, F., E. Viedma, and M. Lozano, Fuzzy tools to improve genetic algorithms, in Zimmermann, H., editor, *Proceedings of the 2nd European Congress on Intelligent Techniques and Soft Computing*, pp. 1532-1539, Aachen, Germany, 1994.

[296] Hesser, J. and R. Männer, Towards an optimal mutation probability for genetic algorithms, in Schwefel and Männer [568], pp. 23-32.

[297] Hightower, R. R., Computational aspects of antibody gene families, Ph. D. dissertation, University of New Mexico, 1996.

[298] Hinterding, R., Mapping, order-independent genes and the knapsack problem, in Fogel [197], pp. 13-17.

[299] Hinterding, R., Z. Michalewicz, and A. Eiben, Adaptation in evolutionary computation: a survey, in Porto[512], pp. 65-69.

[300] Hirsch, W. M. and G. B. Dantzig, The fixed charge problem, *Naval Research Logistics Quarterly*, vol. 15, pp. 413-424, 1968.

[301] Ho, L. L., Simulation of condensed-phase physical, chemical, and biological systems on massively parallel computers, Ph. D. dissertation, Yale University, 1997.

[302] Hocaogle C., Multipath planning using evolutionary computation with specification, Ph. D. dissertation, Rensselaer Polytechnic Institute, 1997.

[303] Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975; MIT Press, Cambridge, MA, 1992.

[304] Holsapple, C., V. Jacob, R. Pakath, and J. Zaveri, A genetics-based hybrid scheduler for

generating static schedules in flexible manufacturing contexts, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, pp. 953-971, 1993.

[305] Horn, J., Multicriterion decision making, in Bäck, T., D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pp. 517-522, Oxford University Press, New York, 1997.

[306] Horn, J., N. Nafpliotis, and D. Goldberg, A niched Pareto genetic algorithm for multiobjective optimization, in Fogel [197], pp. 82-87.

[307] Hoschei, G. C., Tabu search/genetic algorithm hybrid heuristic for solving a master production scheduling problem with sequence-dependent changeover times, Ph. D. dissertation, Colorado School of Mines, 1995.

[308] Hou, E. S., H. Ren and N. Ansari, *Dynamic, Genetic, and Chaotic Programming: Efficient Multiprocessor Scheduling Based on Genetic Algorithms*, 1992.

[309] Houck, C. R., Meta-heuristics for manufacturing problems, Ph. D. dissertation, North Carolina State University, 1996.

[310] Huang, R. J., Detection strategies for face recognition using learning and evolution, Ph. D. dissertation, George Mason University, 1998.

[311] Huang, X., Data-driven description of reservoir petrophysical properties, Ph. D. dissertation, University of Tulsa, 1995.

[312] Hughell, D. A., Simulated adaptive management for timber and wildlife under uncertainty, Ph. D. dissertation, North Carolina State University, 1997.

[313] Hwang, C. and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications*, Springer-Verlage, Berlin, 1981.

[314] Igata, S., Study on estimation of short-time rainy areas, using optimized neural networks by genetic algorithms, Ph. D. dissertation, Muroran Institute of Technology, 1995(in Japanese).

[315] Ignizio, J., *Linear Programming in Single and Multiple-Objective Systems*, Prentice Hall, Upper Saddle River, NJ, 1982.

[316] Ijiri, Y., *Management Goals and Accounting for Control*, North-Holland, Amsterdam, 1965.

[317] Ikonen, I. T., Genetic algorithm for a three-dimensional non-convex bin packing problem, Ph. D. dissertation, University of Louisville, 1998.

[318] Inoue, H., Study on an automatic generation method of fuzzy rules by genetic algorithms, Ph. D. dissertation, Ritsumeikan University, 1997.

[319] Ishibuchi, H. and T. Murata, A multiobjective genetic local search algorithm and its application to flowshop scheduling, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, no. 3, pp. 392-403, 1998.

[320] Ishii, H., H. Shiode, and T. Nishida, Stochastic spanning tree problem, *Discrete Applied Mathematics*, vol. 3, pp. 263-273, 1981.

[321] Jacobs, L. W. and M. J. Brusco, A simulated annealing based heuristic for the set-covering problem, Technical report, Operations Management and Information Systems Dspartment, North Illinois University, 1993.

[322] Jain, L. C. and K. Jain, editors, *Proceedings of the 1998 2nd International Conference on Knowledge-Based Intelligent Electronic Systems*, IEEE Press, Piscataway, NJ, 1998.

[323] Jallas, E., Improved model-based decision support by modeling cotton variability and using evolutionary algorithms, Ph. D. dissertation, Mississippi State University, 1998.

[324] Jan, R. H., Design of reliable networks, *Computers and Operations Research*, vol. 20, no. 1, pp. 25-34, 1993.

[325] Jan, R. H., F. J. Hwang, and S. T. Chen, Topological optimization of a communication network subject to a reliability constraint, *IEEE Transactions on Reliability*, vol. 42, no. 1, pp. 63-70, 1993.

[326] Jensen, A. P. and J. W. Barnes, *Network Flow Programming*, Wiley, New York, 1980.

[327] Jensen, E. D., Topological structural design using genetic algorithms, Ph. D. dissertation, Purdue University, 1992.

[328] Jensen, R. E., A dynamic programming algorithm for cluster analysis, *Operations Research*, vol. 12, pp. 1034-1057, 1969.

[329] Jeon, Y. C., Genetic algorithm-based non-linear modeling and its application to control and mechanical diagnostics, Ph. D. dissertation, Columbia University, 1994.

[330] Jin, K., Study on optimal neural network design and its application using genetic algorithms, Ph. D. dissertation, Hokkaido University, 1996.

[331] Johnson, D. S., A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, *Journal of SIAM*, vol. 3, pp. 299-325, 1974.

[332] Joines, J. A., Hybrid genetic search for manufacturing cell design, Ph. D. dissertation, North Carolina State Univesity, 1996.

[333] Joines, J. A., Manufacturing cell design using genetic algorithms, M. S. dissertation, North Carolina State University, 1993.

[334] Joines, J. A., C. T. Culbreth, and R. E. King, Manufacturing cell design: an integer programming model employing genetic algorithms, Technical Report NCSU-IE 93-16, North Carolina State University, 1993.

[335] Joines, J. A., C. T. Culbreth, and R. E. King, A genetic algorithm based integer program for manufacturing cell design, *Proceedings of the International Conference on Flexible Automation and Integrated Manufacturing*, Stuttgart, Germany, 1995.

[336] Joines, J. A., C. T. Culbreth, and R. E. King, Manufacturing cell design: an integer Programming model employing genetics, *IIE Transactions*, vol. 28, no. 1, pp. 69-85, 1996.

[337] Joines, J. A., R. E. King, and C. T. Culbreth, Utilizing a hybrid genetic search for manufacturing cell design, *IIE Transactiuons* (in review).

[338] Joines, J. A., R. E. King, and C. T. Culbreth, *Cell formation using genetic algorithms*, in Suresh, N. C. and J. M. Kay, editors, *Group Technology and Cellular Manufacturing*, pp. 185-204, Kluwer Academic Publishers, Norwell, MA, 1998.

[339] Joines, J. and C. Houck, On the use of non-stationary penalty functions to solve nonlinear

constrained optimization problems with GAs, in Fogel[197], pp. 579 584.

[340] Jones, M. H., The Hereford Ranch algorithm: an improvement in genetic algorithms using selective breeding, Ph. D. dissertation, Utah State University, 1995.

[341] Julstrom, B., What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm, in Eshelman[177], pp. 81-87.

[342] Kaiser, T., Dynamic load distributions for adaptive computations on MIMD machines usign hybrid genetic algorithms, Ph. D. dissertation, University of New Mexico, 1997.

[343] kamat, S. J. and M. W. Riley, Determination of reliability using event-based Monte Carlo simulation, *IEEE Transactions on Reliability*, vol. 24, no. 1, pp. 73-75, 1975.

[344] Kang, M. H., Learning evasive maneuvers using evolutionary algorithms and neural networks, Ph. D. dissertation. Mississippi State University, 1997.

[345] Kang, S. and U. Wemmerlov, A work load-oriented heuristic methodology for manufacturing cell formation allowing reallocation of operations, *European Journal of Operational Research*, vol. 69, no. 3, pp. 292-311, 1993.

[346] Kargupta, H., Search, polynomial complexity, and the fast messy genetic algorithm, Ph. D. dissertation, University of Illinois at Urbana-Champaign, 1996.

[347] Kasahara, H. and S. Narita, Parallel processing of robot-arm control computation on a multimicroprocessor system, *IEEE Journal of Robotics and Automation*, vol. 1, no. 2, pp. 104-113, 1985.

[348] Kasilingam, J. A. and R. S. Lashkari, Cell formation in the presence of alternative process plans in FMS, *Production Planning and Control*, vol. 2, no. 2, pp. 135-141, 1991.

[349] Kaufmann, A. and M. Gupta, *Fuzzy Mathematical Models in Engineering and Management Science*, 2nd edition, North-Holland, Amsterdam, 1991.

[350] Kazerooni, M., Cell formation using genetic algorithms, in *Proceedings of the International Conference on Flexible Automation and Integrated Manufacturing*, Stuttgart, Germany, 1995.

[351] Kazerooni, M., L. Luong, and K. Abhary, Cell formation using genetic algorithms, *Proceedings of the International Conference on Flexible Automation and Integrated Manufacturing*, vol. 3, no. 3-4, pp. 283-301, 1996.

[352] keeney, R. L. and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Wile, New York, 1976.

[353] Kennedy, S., Five ways to a smarter genetic algorithm, *AI Expert*, pp. 35-38, 1993.

[354] Kennington, J. L. and V. E. Unger, A new branch and bound algorithm for the fixed charge transportation problem, *Management Science*, vol. 22, no. 10, pp. 1116-1126, 1976.

[355] Kermani, B. G., On using artificial neural networks and genetic algorithms to optimize performance of an electric noise, Ph. D. dissertation, North Carolina State University, 1996.

[356] Kershenbaum, A., Computing capacitated minimal spanning trees efficiently, *Networks*, vol. 4, PP. 299 310, 1974.

[357] Kershenbaum, A., *Telecommunications Network Design Algorithms*, McGraw-Hill, New

York, 1993.

[358] Kershenbaum, A., R. R. Boorstyn, and R. Oppenheim, Centralized teleprocessing network design, *Networks*, vol. 13, pp. 279-293, 1983.

[359] Kershenbaum, A. and R. V. Slyke, Recursive analysis of network reliability, *Networks*, vol. 3, pp. 81-94, 1973.

[360] Khuri, S., T. Bäck, and J. Heitkotter, The zero/one multiple knapsack problem and genetic algorithms, in *Proceedings of the ACM Symposium on Applied Computation*, 1994.

[361] Kim, J. R., M. Gen, and K. Ida, Bicriteria network design using spanning treebased genetic algorithm, *Artificial Life and Robotics*, 1999(forthcoming).

[362] Kim, J. R., M. Gen, and K. Ida, A spanning tree-based genetic algorithm for reliable multiplexed network topology design, in *Proceedings of the 4th International Symposium on Artificial Life and Robotics*, vol. 2, pp. 460-463, 1999.

[363] Kim, J. R. and M. Gen, A GA for bicriteria communication network topology design, *Engineering Valuation and Cost Analysis*, (forthcoming).

[364] Kimura, T., A two-moment approximation for the mean waiting time in the *GI/G/s* queue, *Management Science*, vol. 32, no. 6, pp. 751-763, 1986.

[365] Kimura, T., Approximations for multi-server queues; system interpolations, *Queuing Systems*, vol, 17, pp. 347-382, 1994.

[366] King, J. R., Machine-component grouping formation in group technology, *OMEGA; International Journal of Management Science*, vol. 8, no. 2, pp. 193-199, 1980.

[367] King, J. R., Machine-component grouping in production flow analysis; an approach using a rank order clustering algorithm, *Int. Journal of Production Research*, vol. 18, no. 2, pp. 213-232, 1980.

[368] King, J. R. and V. Nakornchai, Machine-component group formation in group technology; review and extension, *Int. Journal of Production Research*, vol. 20, no. 2, pp. 117-133, 1982.

[369] Klingman, D., A. Naoier, and J. Stutz, A program for generating large scale capacitated assignment, transportation and minimum cost flow networks, *Management Science*, vol. 20, no. 5, pp. 814-822, 1974.

[370] Kobayashi, S., Foundations of genetic algorithms and its applications, *Communications of ORSJ*, vol. 45, pp. 256-261, 1993(in Japanese).

[371] Kobayashi, S., I. Ono, and M. Yamamura, An efficient genetic algorithm for job shop scheduling problems, in Eshelman[177], pp. 506-511.

[372] Koh, S. J. and C. Y. Lee, A tabu search for the survivable fiber optic communication network design, *Computers and Industrial Engineering*, vol. 28, no. 4, pp. 689-700, 1995.

[373] Kommu, V., Enhanced genetic algorithms in constrained search spaces with emphases on parallel environments, Ph. D. dissertation, University of Iowa, 1993.

[374] Kou, S., Decision support for irrigated project planning using a genetic algorithm, Ph. D. dissertation, University of Oklahoma, 1997.

[375] Koza, J. R., *Genetic Programming*, MIT Press, Cambridge, MA, 1992.

[376] Kruska, J. Jr., On the shortest spanning subtree of a graph and the traveling salesman problem, *Proceedings of the ACM*, vol. 7, no. 1, pp. 48-50, 1956.

[377] Kubota, N. and T. Fukuda, Schema representation in virus-evolutionary genetic algorithm for knapsack problem, in Fogel[196], pp. 834-838.

[378] Kumamoto, H., K. Tanaka, and K. Inoue, Efficient evaluation of system reliability by Monte Carlo method, *IEEE Transactions on Reliability*, vol. 26, no. 5, pp. 311-315, 1977.

[379] Kumar, A. R. and Y. P. Gupta, Genetic-algorithm-based reliability optimization for computer network expansion, *IEEE Transactions on Reliability*, vol. 44, no. 1, pp. 63-72, 1995.

[380] Kumar, A. R., M. Pathak, Y. P. Gupta, and H. R. Parsaei, a genetic algorithm for distributed system topology design, *Computers and Industrial Engineering*, vol. 28, no. 3, pp. 659-670, 1995.

[381] Kumar, K. R. and M. P. Chandrasekharan, Grouping efficacy: a quantitative criterion for goodness of block diagonal forms of binary matrices in group technology, *Int. Journal of Production Reseach*, vol. 28, no. 2, pp. 233-243, 1990.

[382] Kuo, W., V. R. Prasad, F. A. Tillman, and C. L. Hwang, *Fundamentals and Applications of Reliability Optimization* (forthcoming).

[383] Kursawa, F., A variant of evolution strategies for vector optimization, in Schwefel and Männer [568], pp. 193-197.

[384] Kusiak, A. and M. Cho, Similarity coefficient algorithms for solving the group technology problem, *Int. Journal of Production Research*, vol. 30, no. 11, pp. 2633-2646, 1992.

[385] Kusiak, A. and W. Chow, Efficient solving of the group technology problem, *OMEGA: International Journal of Management Science*, vol. 6, no. 2, pp. 117-124, 1987.

[386] Kusiak, A. and W. Chow, Decomposition of manufacturing systems, *IEEE Journal of Robotics and Automation*, vol. 4, no. 5, pp. 457-471, 1988.

[387] Kusiak, A. and G. Finke, Selection of process plans in automated manufacturing systems, *IEEE Journal of Robotics and Automation*, vol. 4, no. 4, pp. 397-402, 1988.

[388] Kusiak, A. and S. Heragu, The facility layout problem, *European Journal of Operational Research*, vol. 29, pp. 229-251, 1987.

[389] Labordere, H., Partitioning algorithm in mixed and pseudo mixed integer Programming, Ph. D. dissertation, Rensselaer Polytechnic Institute, 1969.

[390] Lai, Y. and C. Hwang, *Fuzzy Mathematical Programming*, Springer-Verlag, Berlin, 1992.

[391] Larson, C. B., Vibration testing by design: excitation and sensor placement using genetic algorithms, Ph. D. dissertation, University of Florida, 1996.

[392] Lazio, T. J. W., Genetic algorithms, pulsar planets, and ionized interstellar microturbulence, Ph. D. dissertation, Cornell University, 1997.

[393] Lee, A. M. and P. A. Longton, Queuing Process associated with airling passenger check-in, *Operations Research Quarterly*, vol. 10, pp. 56-71, 1957.

[394] Lee, B., Three new algorithms for exact D-optimal design problems, Ph. D. dissertation, Ohio

State University, 1993.

[395] Lee, C., Fuzzy logic in control systems: fuzzy logic controller, part Ⅰ, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, pp. 404-418, 1990.

[396] Lee, C., Fuzzy logic in control systems: fuzzy logic controller, part Ⅱ, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, pp. 419-435, 1990.

[397] Lee, H. and A. Garcia-Diaz, A network flow approach to solve clustering problems in group technology, *Int. Journal of Production Research*, vol. 31, no. 3, pp. 603-612, 1993.

[398] Lee, H. and A. Garcia-Diaz, Network flow procedures for the analysis of cellular manufacturing systems, *IIE Transactions*, vol. 28, pp. 333-345, 1996.

[399] Lee, J., Tolerance optimization using genetic algorithm and approximated simulation, Ph. D. dissertation, University of Michigan, 1992.

[400] Lee, J., Genetic algorithms in multidisciplinary design of low-vibration rotors, Ph. D. dissertation, Rensselaer Polytechnic Institute, 1996.

[401] Lee, M. A. Automatic design and adaptation of fuzzy systems and genetic algorithms using soft computing techniques, Ph. D. dissertation, University of California-Davis, 1994.

[402] Lee, M. and H. Takagi, Dynamic control of genetic algorithm using fuzzy logic techniques, in Forrest[204], pp. 76-83.

[403] Lee, S., *Goal programming for Decision Analysis*, Auerbach Publishers, Philadelphia, 1972.

[404] Leskowsky, L., L. Logan, and vannelli, Modern production management systems, in Kusiak, A., editor, *Group Technology Decision Aides in a Expert System for Plant Layout*, pp. 561-585, North-Holland, Amsterdam, 1987.

[405] Levin, A., Fleet routing and scheduling problems for air transportation systems, Ph. D. dissertation, Massachusetts Institute of Technology, 1969.

[406] Leving, D. M., Parallel genetic algorithm for the set partitioning problem, Ph. D. dissertation, Illinois Institute of Technology, 1994.

[407] Li, D., Multiobjective optimum design of static and seismic-resistant structures with genetic algorithms, fuzzy logic, and game theory, Ph. D. dissertation, University of Missouri-Rolla, 1998.

[408] Li, J., On an algorithm for solving fuzzy linear systems, *Fuzzy Sets and Systems*, vol. 61, pp. 369-371, 1994.

[409] Li, J., Oil tanker market modeling, analysis, and forecasting using neural networks, fuzzy logic, and genetic algorithms, Ph. D. dissertation, University of Michigan, 1997.

[410] Li, Y. X., Multi-criteria optimization techniques and its applications to engineering design problems, M. S. dissertation, Ashikga Institute of Technology, 1996.

[411] Li, Y. Z., Study on hybridized genetic algorithms for production distribution planning problems, Ph. D. dissertation, Ashikaga Institute of Technology, 1999.

[412] Li, Y. Z., and M. Gen, Spanning tree-based genetic algorithm for bicriteria transportation problem with fuzzy coefficients, *Australian Journal of Intelligent Information Processing Systems*, vol. 4, no. 3, pp. 220-229, 1998.

[413] Li, Y. Z. and M. Gen, Spanning tree based genetic algorithm for solving bicriteria transportation problem, *Journal of the Japan Society for Fuzzy Theory and Systems*, vol. 10, no. 5, pp. 888-898, 1998.

[414] Li, Y. Z., M. Gen, and K. Ida, Improved genetic algorithm for solving multiobjective solid transportation problem with fuzzy numbers, *Japanese Journal of Fuzzy Theory and Systems*, vol. 9, no. 2, pp. 239-250, 1997.

[415] Li, Y. H., Genetic algorithm-based design of multiplierless two-dimensional state-space digital filters, Ph. D. dissertation, Tohoku University, 1996.

[416] Lin, S. C., Genetic algorithm-based scheduling system for dynamic job shop scheduling problem, Ph. D. dissertation, Michigan State University, 1997.

[417] Lin, W., High-quality tour hybrid genetic schemes for TSP optimization problems, Ph. D. dissertation, State University of New York-Binghamton, 1995.

[418] Liu, B., *Uncertain Programming*, Wiley, New York, 1999.

[419] Liu, B. and A. O. Esogbue, *Decision Criteria and Optimal Inventory Processes*, Kluwer Academic Publishers, Boston, 1999.

[420] Liu, C., M. Dai, X. Y. Wu, and W. K. Chen, A network overall reliability algorithm, in *Proceedings of Neural, Parallel and Scientific Computations*, pp. 287-292, Atlanta, GA, 1995.

[421] Liu, C. and J. Wu, Machine cell formation: using the simulated annealing algorithm, *Computer Integrated Manufacturing*, vol. 6, no. 6, pp. 335-349, 1993.

[422] Liu, D. G., Application of multiobjective genetic algorithms to control systems design, Ph. D. dissertation, Tohoku University, 1996.

[423] Loh, J. D., Automated discovery of self-replicating structures in cellular space automata models, Ph. D. dissertation, University of Maryland, 1996.

[424] Liou, T. S. and M. J. Wang, Ranking fuzzy numbers with integral value, *Fuzzy Sets and systems*, vol. 50, pp. 247-255, 1992.

[425] Louveaux, F. V., Stochastic location analysis, *Location Science*, vol. 1, no. 2, pp. 127 154, 1993.

[426] Lu, B. L., Study of genetic algorithms on a application to industrial design, Ph. D. dissertation, Muroran Institute of Technology, 1996.

[427] Ludvig, J., J. Hesser, and R. Manner, Tackling the representation problem by stochastic averaging, in Bäck[29], pp. 196-203.

[428] Luenberger, D., *Linear and Nonlinear Programming*, 2nd edition, Addison-Wesley, Reading, MA, 1984.

[429] Luhandjula, M. K., Fuzzy optimization: an appraisal, *Fuzzy Sets and Systems*, vol. 30, pp. 257-282, 1989.

[430] Luhandjula, M., Linear programming under randomness and fuzziness, *Fuzzy Sets and Systems*, vol. 10, pp. 45-55, 1983.

[431] Luhandjula, M., On possibilistic linear programming, *Fuzzy Sets and Systems*, vol. 18, pp.

1530, 1986.

[432] Mahfoud, S., Niching methods for genetic algorithms, Ph. D. dissertation. University of Illinois at Urbana-Champaign, 1995.

[433] Maifeld, T. T., Genetic-based unit commitment algorithm, Ph. D. dissertation, Iowa State University, 1997.

[434] Malik, K. and G. Yu, A branch and bound algorithm for the capacitated minimum spanning tree problem, *Networks*, vol. 23, pp. 525-532, 1993.

[435] Mandaltsis, D. and J. M. Kontolen, Overall reliability determination of computer networks with hierarchical routing strategies, *Microelectronics and Reliability*, vol. 27, no. 1, pp. 129-143, 1987.

[436] Mantawy, A. H., Unit commitment by artificial intelligence techniques, Ph. D. dissertation, King Fahd University of Petroleum and Minerals(Saudia Arabia), 1997.

[437] Marchelya, J. C., Enforcing pollution control laws: Stackleberg-Markov game models for improving efficiency and effectiveness, Ph. D. dissertation, Johns Hopkins University, 1997.

[438] Maria, A., Genetic algorithm for multimodel continuous optimization problems, Ph. D. dissertation, University of Oklahoma, 1995.

[439] Martello, S. and P. Toth, The 0-1 knapsack problem, in Christofides, N., A. Mingozzi, and C. Sandi, editors, *Combinatorial Optimization*, pp. 237-279. Wiley, Chichester, West Sussex, England, 1979.

[440] Martello, S. and P. Toth, Algorithms for knapsack problems, *Annals of Discrete Mathematics*, vol, 31, pp. 213-257, 1987.

[441] Martello, S. and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, Chichester, West Sussex, England, 1990.

[442] Martins, E., On a multicriteria shortest path problem, *European Journal of Operational Research*, vol. 16, pp. 236-245, 1984.

[443] Masters, B. P., Evolutionary design of controlled structures, Ph. D. dissertation, Massachusetts Institute of Technology, 1997.

[444] Matsui, K., Expression of genetic algorithms to macroadaptation degree and its application to picture processing systems, Ph. D. dissertation, Tokyo Institute of Technology, 1996.

[445] Matthew, R. M., Applications and limitations of genetic algorithms for the optimization of multivariate calibration techniques, Ph. D. dissertation, Clemson University, 1998.

[446] McAuley, J., Machine grouping for efficient production, *Production Engineering*, vol. 51, no. 2, pp. 53-57, 1972.

[447] McCormick, W. T., P. J. Schweitzer, and T. W. White, Problem decomposition and data reorganization by a cluster technique, *Operations Research*, vol. 20, no. 5, pp. 993-1009, 1972.

[448] Meddin, M., Genetic algorithms: a Markov chain and detail balance approach, Ph. D. dissertation, Ggorgia Institute of Technology, 1995.

[449] Memon, G. Q., Optimization methods for real-time traffic control, Ph. D. dissertation,

University of Pittsburgh, 1995.

[450] Merkle, L. D., Analysis of linkage-friendly genetic algorithms, Ph. D. dissertations, Air Force Institute of Technology, 1996.

[451] Michael, F., Form-finding analysis of vibrating towered shells of revolution as an inverse eigenproblem and as a genetic algorithm optimization problem, Ph. D. dissertation, University of Tokyo, 1996.

[452] Michalewicz, Z., Genetic algorithms, numerical optimization, and constraints, Eshelman[170], pp. 151-158.

[453] Michalewicz, Z., A survey of constraint handling techniques in evolutionary computation methods, In McDonnell, J., R. Reynolds, and D. Fogel, editors, *Proceedings of the 4th Annual Conference on Evolutionary programming*, pp. 135-155, MIT Press, Cambridge, MA, 1995.

[454] Michalewicz, Z., Evolutionary computation: practical issues, in Fogel[195], pp. 30-39.

[455] Michalewicz, Z., *Genetic Algorithm + Data Structure = Evolution Programs*, 3rd edition, Springer-Verlag, New York, 1996.

[456] Michalewicz, Z., T. Logan, and S. Swaminathan, Evolutionary operators for continuous convex parameter spaces, in Sebald, A. and L. Fogel, editors, *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pp. 84-97, World Scientific Publishing, River Edge, NJ, 1994.

[457] Michalewicz, Z., G. A. Vignaux, and M. Hobbs, A non-standard genetic algorithm for the nonlinear transportation problem, *ORSA Journal of Computing*, vol. 3, no. 4, pp. 307-316, 1991.

[458] Mock, K. J., Intelligent information filtering via hybrid techniques: hill climbing, case-based reasoning, index patterns, and genetic algorithms, Ph. D. dissertation, University of California-Davis, 1996.

[459] Modi, A. K. and I. A. Karimi, Design of multiproduct batch processes with finite intermediate storage, *Computers and Chemical Engineering*, vol. 13, pp. 127-139, 1989.

[460] Monem, M. J., Performance evaluation and optimization of irrigation canal system using genetic algorithm, Ph. D., dissertation, University of Calgary, 1996.

[461] Monma, C. L. and C. N. Potts, On the complexity of scheduling with batch setup times, *Operations Research*, vol. 37, pp. 798-804, 1989.

[462] Moon, B. R. and C. K. Kim, A two-dimensional embedding of graphs for genetic algorithms, in Back[29], pp. 204-211.

[463] Moon, C. and C. K. Kim, and M. Gen, Genetic algorithm for maximizing the parts flow within cells in manufacturing cell design, *Computers and Industrial Engineering*, vol. 2, pp. 1730-1733, 1999.

[464] Moon, C. and M. Gen, A genetic algorithm-based approach for design of independent manufacturing cells, *International Journal of Production Economics*, vol. 60-61, pp. 421-426, 1999.

[465] Moon, C., M. Gen, and A. Suer, A genetic algorithm-based model for minimizing additional capital investment in manufacturing cell design, *Engineering Valuation and Cost Analysis*, vol. 2, pp. 133-142, 1999.

[466] Moon, Y., High-performance simulation-based optimization environment for large-scale systems, Ph. D. dissertation, University of Arizona, 1996.

[467] Morikawa, K., Study on scheduling using genetic algorithms, Ph. D. dissertation, Nagoya University, 1996.

[468] Moscato, P. and M. Norman, A memetic approach for the travelling salesman problem: implementation of a computational ecology for combinatorial optimization on message-passing systems, in *Proceedings of the International Conference on Parallel Computing and Transputer Applications*, Amsterdam, 1992.

[469] Moy, J., *RFC-1131*: *The OSPF Specification*, 2nd edition, Network Working Group, New York, 1989.

[470] Mühlenbein, H., M. Schlouter, and O. Kraemer, Evolution algorithms in combinatorial optimization, *Parallel Computing*, vol. 7, pp. 65-85, 1988.

[471] Mulvey, J. and H. Crowder, Cluster analysis: an application of Lagrangian relaxation, *Management Science*, vol. 25, pp. 329-340, 1979.

[472] Munakata, T. and D. J. Hashier, A genetic algorithm applied to the maximum flow problem, in Forrest[204], pp, 488-493.

[473] Munemoto, M., Study on disperse control-type dynamic load balance using genetic algorithms, Ph. D. dissertation, Hokkaido University, 1995.

[474] Munetomo, M., Y. Takai, and Y. Sato, An adaptive network routing algorithm employing path genetic operators, in Bäck[29], pp. 643-649.

[475] Murata, T., Genetic algorithms for multiobjective optimization, Ph. D. dissertation, University of Osaka Prefecture, 1996.

[476] Murata, T., H. Ishibuchi, and H. Tanaka, Multiobjective genetic algorithm and its application to flowshop scheduling, *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 957-968, 1996.

[477] Murthy, J. and D. Olson, An interactive procedure using domination cones for bicriterion shortest path problems, *European Journal of Operational Research*, vol. 72, pp. 417-431, 1994.

[478] Murty, K., *Linear and Combinatorial Programming*, Wiley, New York, 1976.

[479] Nagi, R., G. Harhalakis, and J. M. Proth, Multiple routings and capacity considerations in group technology applications, *Int. Journal of Production Research*. vol. 28, no. 12, pp. 2243-2257, 1990.

[480] Nakamura, M., Study on protocol design and combinatorial optimization in resource assignment problems on the basis of net theory and genetic algorithms, *Ph. D. dissertation*, *Osaka* University, 1995.

[481] Nakanishi, Y., Optimization of a structure phase by homology theory and genetic algorithms,

Ph. D. dissertation, University of Tokyo, 1994.

[482] Narula, S. and C. Ho, Degree-constrained minimum spanning tree, *Computers and Operations Research*, vol. 7, pp. 239-249, 1980.

[483] Ng,S., Worst-case analysis of an algorithm for cellular manufacturing, *European Journal of Operational Research*, vol. 69, no. 3, pp. 384-398, 1993.

[484] Niesse,J. A., Structural optimization of atomic and molecular microclusters using a genetic algorithm in real-valued space-fixed coordinates, Ph. D. dissertation, University of New Hampshire, 1998.

[485] Nims,J. W., Contingency ranking for voltage stability using a genetic algorithm, Ph. D. dissertation, University of Alabama, 1995.

[486] Noga, A. J., Performance analysis and simulation of a class of numerical demodulation techniques for large-deviation FM signals, Ph. D. dissertation, Syracuse University, 1995.

[487] Nolte, B. J., Global optimization algorithms for seismic inversion, Ph. D. dissertation, University of Hawaii, 1995.

[488] Oei,C. K., D. E. Goldberg, and S. J. Chang, Tournament selection, niching, and the preservatin of diversity, Technical report, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 1991.

[489] Oh,K., On-line optimization of substrates feeding in a hybrid cell vulture using an artificial neural network and a genetic algorithm, Ph. D. dissertation, Osaka University, 1996.

[490] Oliver, I., D. Smith, and J. Holland, A study of permutation crossover operators on the traveling salesman problem, in Grefenstette[267], pp. 224-230.

[491] Olsen,A., Penalty functions and the knapsack problem, in Fogel[197], pp. 554-558.

[492] Ono,I. and S. Kobayashi, A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover, in Bäck[29], pp. 246-253.

[493] Ono,I., M. Yamamura, and S. Kobayashi, A genetic algorithm for job-based order crossover, in Fogel[195], pp. 547-552.

[494] Orvosh, D. and L. Davis, Using a genetic algorithm to optimize problems with feasibility constraints, in Fogel[197], pp. 548-552.

[495] Osyczka,A. and S. Kundu, A new method to solve generalized multicriterion optimization problems using genetic algorithm, *Structural Optimization*, vol. 10, no. 2, pp. 94-99, 1995.

[496] Osyczka, A. and S. Kundu, A modified distance method for multicriterion optimization using genetic algorithm, *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 871-882, 1996.

[497] Palazolo, P. J., Use of genetic algorithms in bounded search for design of biological nitrification/denitrification waste treatment systems, Ph. D. dissertation, George Mason University, 1997.

[498] Palekar, U. S., M. H. Karwan, and S. Zionts, Approaches for solving fixed charge transportation problem, Ph. D. dissertation, State University of New York-Buffalo, 1986.

[499] Palmer, C. C., Approach to a problem in network design using genetic algorithms, Ph. D. dissertation, Polytechnic University, 1994.

[500] Palmer, C. C. and A. Kershenbaum, An approach to a problem in network design usign genetic algorithms, *Networks*, vol. 26, pp. 151-163, 1995.

[501] Panwalkar, S. and W. Iskander, A survey of scheduling rules, *Operations Research*, vol. 25, pp. 45-61, 1977.

[502] Papadimitriou, C. H., The complexity of the capacitated tree problem, *Networks*, vol. 8, pp. 217-230, 1978.

[503] Pareto, V., *Manuale di Econòmica Polittica*, Società Editrice Libràia, Milan, Italy, 1906; translated into English by A. S. Schwier, as *Manual of Political Economy*, Macmillan, New York, 1971.

[504] Parrill, A. L., Applications of artificial intelligence in drug design, Ph. D. dissertation, University of Arizona, 1996.

[505] Patel, H. C., Genetic algorithms: a tool for quantitative structure-activity relationship analyses, Ph. D. dissertation, University of Illinois-Chicago, 1996.

[506] Patterson, J. and G. Roth, Scheduling a project under multiple resource constraints: a 0-1 programming approach, *AIIE Transactions*, vol. 8, pp. 449-455, 1976.

[507] Peterson, L. L. and B. S. Davie, *Computer Networks: A Systems Approach*, Morgan Kaufmann Publishers, San Francisco, 1996.

[508] Pheatt, C. B., Construction of D-optimal experimental designs using genetic algorithms, Ph. D. dissertation, Illinois Institute of Technology, 1995.

[509] Picardo, L., Framework for the analysis of evolutionary algorithms, Ph. D. dissertation, Case Western Reserve University, 1996.

[510] Pierre, S., M. Hyppolite, J. Bourjolly, and O. Dioume, Topological design of computer communication networks using simulated annealing, *Engineering Applications of Artificial Intelligence*, vol. 8, no. 1, pp. 61-69, 1995.

[511] Pinon, E., Investigation of the applicability of genetic algorithms to spacecraft trajectory optimization, Ph. D. dissertation, University of Texas—Austin, 1995.

[512] Porto, B., editor, *Proceedings of the IEEE International Conference on Evolutionary Computation*, IEEE Press, Piscataway, NJ, 1997.

[513] Potter, M. A., Design and analysis of a computational model of cooperative coevolution, Ph. D. dissertation, George Mason University, 1997.

[514] Potts, C. N. and L. V. Wassenhove, Integrating scheduling with batch and lotsizing: a review of algorithms and complexity, *Journal of the Operations Research Society*, vol. 43, pp. 395-406, 1992.

[515] Prim, R., Shortest connection networks and some generalizations, *Journal of Bell Systems Technology*, vol. 36, pp. 1389-1401, 1957.

[516] Pritsker, A., L. Waters, and P. Wolfe, Multiproject scheduling with limited resources: a 0-1 approach, *Management Science*, vol. 16, pp. 93-108, 1069.

[517] Prüfer, H., Neuer Beweis eines Satzes über Permutationen, *Archiv fuer Mathematische und Physik*, vol. 27, pp. 742-744, 1918.

[518] Psaraftis, H. N., A dynamic programming approach for sequencing groups of identical jobs, *Operations Research*, vol. 28, pp. 1347-1359, 1980.

[519] Qiao, Z., Y. zhang, and G. Wang, On fuzzy random linear programming, *Fuzzy Sets and Systems*, vol. 65, pp. 31-49, 1994.

[520] Radcliffe, N. and P. Surry, Formal memetic algorithms, in Fogarty[194], pp. 1-16.

[521] Rai, S. and D. Agrawal, editors, *Distributed Computing Network Reliability*, IEEE Computer Society Presss, Los Alamitos, CA, 1990.

[522] Raidl, G. R., An improved genetic algorithm for the multiconstrained 0-1 knapsack problem, in Fogel[196], pp. 207-211.

[523] Rajagopalan, R. and J. L. Batra, Design of cellular production systems: a graph-theoretic approach, *Int. Journal of Production Research*, vol. 13, no. 6, pp. 567-579, 1975.

[524] Rajyabaquse, R. A., Fuzzy logic application of genetic algorithm techniques for adaptive control of nonlinear systems, Ph. D. dissertation, University of Tokyo, 1998.

[525] Ramakumar, R., *Engineering Reliability: Fundamentals and Applications*, Prentice Hall, Upper Saddle River, NJ, 1993.

[526] Rao, S. S., *Engineering Optimization: Theory and Practice*, Wiley, New York, 1996.

[527] Rasheed, K. M., GADO: a genetic algorithm for continuous design optimization, Ph. D. dissertation, University of New Jersey—New Brunswick, 1998.

[528] Ravindran, A., D. T. Phillips, and J. J. Solberg, *Operations Research: Principles and Practice*, Wiley, New York, 1987.

[529] Rawlins, G., editor, *Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, San Francisco, 1991.

[530] Rechenberg, I., *Evolutionstrategie: Optimieriung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart, Germany, 1973.

[531] Reeves, C., Diversity and diversification in genetic algorithms: some connections with tabu search, in Albrecht, R., C. Reeves, and N. Steele, editors, *Artificial Neural Nets and Genetic Algorithms*, pp. 344-351, Springer-Verlag, New York, 1993.

[532] Reklaitis, G. V., A. Ravindran, and K. M. Ragsdell, *Engineering Optimization: Methods and Applications*, Wiley, New York, 1983.

[533] Renders, J. and H. Bersini, Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways, in Fogel[1977], pp. 312-317.

[534] Requicha, A. G. and J. Vandenbrande, Automated systems for process planning and part programming, in *Artificial Intelligence: Implication for CIM*, pp. 301-326, Kempston, Bedfordshire, England, 1988.

[535] Revelle, C., D. Marks, and J. Liebman, An analysis of private and public sector location models, *Management Science*, vol. 16, no. 11, pp. 692-707, 1970.

[536] Rho, S., Distributed database design: allocation of data and operations to nodes in distributed database systems, Ph. D. dissertation, University of Minnesota, 1995.

[537] Richardson, J., M. Palmer, G. Liepins, and M. Hilliard, Some guidelines for genetic

algorithms with penalty functions. in Schaffer[564], pp. 191-197.

[538] Rocha,L., Evidence sets and contextual genetic algorithms: exploring uncertainty, context, and embodiment in cognitive and biological systems, Ph. D. dissertation, State University of New York—Binghamton, 1997.

[539] Rogers,L. L., Optimal groundwater remediation using artificial neural networks and a genetic algorithm, Ph. D. dissertation, Stanford University, 1992.

[540] Root,J., An application of symbolic logic to a selection problem, *Operations Research*, vol. 12, no. 4, pp. 519-526, 1964.

[541] Ros,J. P., Learning Boolean functions with genetic algorithms: a PAC analysis, Ph. D. dissertation, University of Pittsburgh, 1992.

[542] Rosin, C., Coevolutionary search among adversaries, Ph. D. dissertation, University of California—San Diego, 1997.

[543] Rankin, R. P., Consideration of rapidly converging genetic algorithms designed for application to problems with expensive evaluation functions, Ph. D. dissertation, University of Missouri—Rolla, 1993.

[544] Rubin,J., A technique for the solution of massive set covering problems with applications to airline crew scheduling, *Transportation Science*, vol. 4, pp. 34-48, 1973.

[545] Sá, G., Branch and bound approximate solutions to the capacitated plant location problem, *Operations Research*, vol. 17, no. 6, pp. 1006-1016, 1969.

[546] Sage,A. P., *Systems Engineering*, Wiley, New York, 1992.

[547] Sakawa, M., *Fuzzy Sets and Interactive Multiobjective Optimization*, Plenum Press, New York, 1993.

[548] Sakawa, M., H. Ishii, and I. Nishizaki, *Soft Optimization*, Asakura Publishing, Tokyo, 1995(in Japanese).

[549] Sakawa, M., K. Kato, H. Sunada, and T. Shibano, Fuzzy programming for multiobjective 0-1 programming problems through revised genetic algorithms, *European Journal of Operational Research*, vol. 97, pp. 149-158, 1997.

[550] Sakawa, M., I. Nishizaki, and M. Hitaka, Interactive fuzzy programming for multievel 0-1 programming problems through genetic algorithms, *European Journal of Operational Research*, vol. 114, pp. 580-588, 1999.

[551] Sakawa,M. and T. Shibano, Interactive fuzzy programming for multiobjective 0-1 programming problems through genetic algorithms with double strings, in Da Ruan, editor, *Fuzzy Logic Foundations and Industrial Applications*, pp. 111-128, Kluwer Academic Publishers, Norwell, MA, 1996.

[552] Sakawa, M. and T. Shibano, Multiobjective fuzzy satisficing methods for 0-1 knapsack problems through gentic algorithm, in W. Pedrycz, editor, *Fuzzy Evolutionary Computation*, pp. 157-177, Kluwer Academic Publishers, Norwell. MA, 1997.

[553] Sakawa, M. and H. Yano, An interactive fuzzy satisficing method using augmented minimax problems and its application to environmental systems, *IEEE Transactions on Systems, Man*

and Cybernetics, vol. 15, pp. 720-728, 1985.

[554] Salkin, C. A. and D. De Kluyver, The knapsack problem: a survey, *Naval Research Logistics Quarterly*, vol. 22, pp. 127 144, 1975.

[555] Salkin, H. and J. Saha, Set covering: uses, algorithms and results, Technical report 272, Department of Operations Research, Case Western Reserve University, 1973.

[556] Salkin, H. and K. Mathur, *Foundations of Integer Programming*, North-Holland, New York, 1989.

[557] Salverson, M., The assembly line balancing problem, *Journal of Industrial Engineering*, vol. 6, no. 1, pp. 18-25, 1955.

[558] Sancho, N. G., A multi-objective routing problem, *Engineering Optimization*, vol. 10, pp. 71-76, 1986.

[559] Sarma, J. A., Analysis of decentralized and spatially distributed genetic algorithms, Ph. D. dissertation, University of New Hampshire, 1998.

[560] Sasaki, M., Studies on solution methods for fuzzy reliability design problems by hybrid genetic algorithms, Ph. D. dissertation, Hakkaido University, 1999(in Japanese).

[561] Sato, H., TA proposal and analysis on the alternation of generation models of genetic algorithms, Ph. D. dissertation, Tokyo Institute of Technology, 1996.

[562] Savelsbergh, M., Local search for routing problem with time windows, *Annals of Operations Research*, vol. 4, no. 23, pp. 285-305, 1985.

[563] Schaffer, J., Multiple objective optimization with vector evaluated genetic algorithms, in Grefenstette[266], pp. 93-100.

[564] Schaffer, J., editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Francisco, 1989.

[565] Schoenauer, M. and Z. Michalewicz, Boundary operators for constrained parameter optimization problems, in Bäck[29], pp. 322-329.

[566] Schwefel H., *Numerical Optimization of Computer Models*, Wiley, Chichester, West Sussex, England, 1981.

[567] Schwefel, H., *Evolution and Optimum Seeking*, Wiley, New York, 1995.

[568] Schwefel, H. and R. Männer, editors, *Parallel Problem Solving from Nature*, Sringer-Verlag, New York, 1990.

[569] Seifoddini, H. and P. M. Wolfe, Application of the similarity coefficient method in group technology, *IIE Transactions*, vol. 18, no. 3, pp. 271-277, 1986.

[570] Sen, S., Minimal cost set covering using probabilistic methods, in *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, *pp.* 157-164, 1993.

[571] Sendhoff, B., M. Kreuts, and W. Seelen, A condition for the genotypephenotype mapping: casualty, in Bäck[29], pp. 354-361.

[572] Shaefer, C., The ARGOT strategy: adaptive representation genetic optimizer technique, *in* Grefenstette[267], pp. 50-55.

[573] Shahooker, K. D., Application of the genetic algorithm for computer-aided design of VLSI

layout, Ph. D. dissertation, University of Michigan, 1994.

[574] Shtub, A., Modeling group technology cell formation as a generalized assignment problem, *Int. Journal of Production Research*, vol. 27, no. 5, pp. 775-782, 1989.

[575] Shu, L., Impact of data structures on the performance of genetic algorithm-based learning, Ph. D. dissertation, University of Alberta, 1992.

[576] Skiena, S., *Implementing Discrete Mathematics Combinatorics and Graph Theory with Mathematics*, Addison-Wesley, Reading, MA, 1990.

[577] Smith, J. B. and D. Shier, An empirical investigation of some bicriterion shortest path algorithms, *European Journal of Operational Research*, vol. 43, pp. 216-224, 1989.

[578] Sniedovich, M., A multi-objective routing problem revisited, *Engineering Optimization*, vol. 13, pp. 99-108, 1988.

[579] Sollin, M., Le Trace de canalisation, in Berge, C. and A. Ghouilla-Houri, editors, *Programming, Games, and Transportation Networks*, Wiley, New York, 1965.

[580] Spears, W. M., Role of imitation and recombination in evolutionary algorithms, Ph. D. dissertation, George Mason University, 1998.

[581] Srinivas, N. and K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation*, vol. 2, no. 3, pp. 221-248, 1995.

[582] Srinivasan, G., T, Narendran, and B. Mahadevan, An assignment model for the part-families problem in group technology, *Int. Journal of Production Research*, vol. 28, no. 1, pp. 145-152, 1990.

[583] Stadler, W., A survey of multicriteria optimization or the vector maximization problem: I. 1776-1960, *Journal of Optimization Theory and Applications*, vol. 69, pp. 1-52, 1979.

[584] Stadler, W., A comprehensive bibliography on multicriteria decision making and related areas, Technical report, University of California -Berkeley, 1981.

[585] Starns, G. K., Optimal synthesis of a planar four-bar mechanism with prescribed timing using generalized reduced gradient, simulated annealing and genetic algorithms, Ph. D. dissertation, Iowa State University, 1996.

[586] Steinberg, D. I., The fixed charged problem, *Naval Research Logistics Quarterly*, vol. 17, no. 2, pp. 217-236, 1970.

[587] Steinmann, H. and R. Schwinn, Computational experience with a zero-one programming problem, *Operations Research*, vol. 17, no. 5, pp. 917-920, 1969.

[588] Steuer, R. E., *Multiple Criteria Optimization: Theory, Computation, and Application*, Wiley, New York, 1986.

[589] Stinson, J., E. Davis, and B. Khumawala, Multiple resource constrained scheduling using branch and bound, *AIIE Transactions*, vol. 10, pp. 252-259, 1978.

[590] Storer, R., S. Wu, and R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, *Management Sciences*, vol. 38, no. 10, pp. 1495-1510, 1992.

[591] Streifel, R. J., Application of computational intelligence to electromechanical systems, Ph. D.

dissertation, University of Washington, 1996.

[592] Stumpf, J. D., Studies on enhanced operator-oriented genetic algorithms, Ph. D. dissertation, Marquette University, 1996.

[593] Süer, G., Evolutionary programming for designing manufacturing cells, *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 379-384, 1997.

[594] Süer, G., R. Vazquez, Y. Pena, and M. Cortes, Evolutionary programming for part family formation, in Gen, M. et al eds. *Proceedings of the First Japan—USA Joint Workshops on Intelligent Manufacturing Systems*, Ashikaga, pp. 29-41, 1998.

[595] Süer, G., R. Vazquez, Y. Pena, and M. Cores, Cellular design by using evolutionary programming as a tool, in Katai, O., M. Gen, et al eds. *Proceedings of the 2nd Japan—Australia Joint Workshops on Intelligent Evolutionary Systems*, Kyoto, pp. 379-384, 1998.

[596] Süer, G., R. Vazquez, and Y. Pena, Evolutionary Programming in cellular design, Presented at INFORMS Spring Meetings, Cincinnati, 1999.

[597] Süer, G., M. Gen, and C. Moon, Evolutionary programming for designing and controlling manufacturing cells, in *Evolutionary Computations and Intelligent Systems*, Gordon and Breach New York(forthcoming).

[598] Sun, D. L. and R. Batta, Cell formation using tabu search, *Computers and Industrial Engineering*, vol. 28, no. 4, pp. 485-494, 1995.

[599] Sun, M., J. E. Aronson, P. G. McKeown, and D. Drinka, A tabu search heuristic procedure for the fixed charge transportation problem, *European Journal of Operational Research*, vol. 106, pp. 441-456, 1998.

[600] Sun, R., Evolving population-based search algorithms through thermodynamic operation: dynamic system design and integration, Ph. D. dissertation, University of Maryland, 1995.

[601] Sweeney, D. J. and R. A. Murphy, Branch-and-bound methods for multi-item scheduling, *Operations Research*, vol. 29, pp. 853-864, 1981.

[602] Swets, D. L., Self-organizing hierarchical optimal subspace learning and inference framework for view-based object recognition and image retrieval, Ph. D. dissertation, West Virginia University, 1996.

[603] Syswerda, G., Uniform crossover in genetic algorithms, in Schaffer[564], pp. 2-9.

[604] Syswerda, G., Scheduling optimization using genetic algorithms, in Davis[147], pp. 332-349.

[605] Tadikonda, S. K., Automated image segmentation and interpretation using genetic algorithms and semantic nets, Ph. D. dissertation, University of Iowa, 1993.

[606] Tagami, T., Study on application method of genetic algorithms for combinatorial optimization problems, Ph. D. dissertation, University of Tokushima, 1995.

[607] Taguchi, T., K. Ida, and M. Gen, Method for solving nonlinear goal programming with interval coefficients using genetic algorithms, *Computers and Industrial Engineering*, vol. 33, no. 1-4, pp. 579-600, 1997.

[608] Taha, H. A., *Integer Programming*, Academic Press, San Diego, CA, 1975.

[609] Takagi, T., Study on an automatic structure method using a fuzzy reasoning controller with

neural networks and genetic algorithms, Ph. D. dissertation, Tokai University, 1994.

[610] Takashi, K. , Study of evolutional mechanisms of cooperative problem solving: a hybrid genetic algorithm for multialgorithm problems, Ph. D. dissertation, Keio University, 1995.

[611] Talbot, F. and J. Patterson, An efficient integer programming algorithm with network cuts for solving resource-constraints scheduling problems, *Management Science*, vol. 24, pp. 1163-1174, 1978.

[612] Tamaki, H. , H. Kita, and S. Kobayashi, Multiobjective optimization by genetic algorithms: a review, in Fogel[195], pp. 517-522.

[613] Tanaka, H. and K. Asia, Fuzzy solution in fuzzy linear programming, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 1, pp. 325-328, 1984.

[614] Tanenbaum, A. S. , editor, *Computer Networks*, Prentice Hall, Upper Saddle River, 1996.

[615] Tang, A. , Genetic algorithms for optimal operation of soil aquifer treatment systems, Ph. D. dissertation, Arizona State University, 1997.

[616] Tang, J. and D. Wang, An interactive approach based on a GA for a type of quadratic programming problem with fuzzy objective and resources, *Computers and Operations Research*, vol. 24, pp. 413-422, 1997.

[617] Tay, E. H. , Automated generation and analysis of dynamic system designs, Ph. D. dissertation, Massachusetts Institute of Technology, 1995.

[618] Terano, T. , K. Asai, and M, Sugeno, *Applied Fuzzy Systems*, Academic Press, London, 1994.

[619] Thierens, D. and D. Goldberg, Convergence models of genetic algorithm selection schemes, in Davidor et al. [142], pp. 119-129.

[620] Tillman, F. , C. Hwang, and W. Kuo, *Optimization of Systems Reliability*, Marcel Dekker, New York, 1980.

[621] Timothy, S. J. , Optimization of sequencing problems using genetic algorithms, Ph. D. dissertation, Colorado State University, 1993.

[622] Tiwari, R. N. , S. Dharmar, and J. R. Rao, Priority structure in fuzzy goal programming, *Fuzzy Sets and Systems*, vol. 19, pp. 251-259, 1986.

[623] Tom, A. and A. Zilinskas, *Global Optimization*, Springer-Verlag, New York, 1989.

[624] Tomasz, O. , Nonlinear adaptive filtering: the genetic algorithm approach, Ph. D. dissertation, Politechnika Warszawska, 1995.

[625] Tompkins, G. H. , Optimization of qualitative variables in discrete event simulation models, Ph. D. dissertation, Kansas State University, 1995.

[626] Toregas, C. , The location of emergency service facilities, *Operations Research*, vol. 19, no. 6, pp. 1363-1373, 1971.

[627] Tsujimura, Y. and M. Gen, Genetic algorithms for solving multiprocessor scheduling problems, in Yao, X. , J. H. Kim, and T. Furuhashi, editors, *Simulated Evolution and Learning*, pp. 106-115, Springer-Verlag, Heidelberg, 1997.

[628] Tsujimura, Y. and M. Gen, Entropy-based genetic algorithm for solving TSP, in Jain and Jain

[322], pp. 285-290.

[629] Tsujimura, Y., M. Gen, and R. W. Cheng, Improved genetic algorithms for job-shop scheduling problems, *Engineering Design and Automation*, vol. 3, no. 2, pp. 133-144, 1997.

[630] Tsujimura, Y., M. Gen, R. Cheng, and T. Momota, Comparative studies on encoding methods of GA for open shop scheduling, *Australian Journal of Intelligent Information Processing Systems*, vol. 4, no. 3-4, pp. 214-219, 1997.

[631] Tsujimura, Y., M. Gen, and D. Zheng, GA-based approach for fuzzy multipleitem inventory control, *Engineering Valuation and Cost Analysis*, vol. 2, pp. 151-157, 1999.

[632] Tsujimura, Y., J. H. Kim, and M. Gen, GA-based design of star-ring LAN topology, in Hwang, H. and N. Tawara, editors, *Proceedings of the First Korea -Japan Joint Conference on Industrial Engineering and Management*, pp. 206-210, Taejon, 1998.

[633] Tsuijimura, Y., S. B. Ko, and M. Gen, Data distribution considered communication flow in network using genetic algorithm, in Jain and Jain[322], pp. 264-271.

[634] Tsujimura, Y. and M. Gen, Parts loading scheduling in a flexible forging machine using an advanced genetic algorithm, *Journal of Intelligent Manufacturing*, vol. 10, no. 2, pp. 149-159, 1999.

[635] Tuson, A. and P. Ross, Cost based operator rate adaptation: an investigation, in Voigt, H., W. Ebelling, I. Rechenberg, and H. Schwefel, editors, *Proceedings of the 4th International Conference on Paraller Problem Solving from Nature*, pp. 461-469, Springer-Verlag, Berlin, 1996.

[636] Valenta, J., Capital equipment decisions: a model for optimal systems interfacing, Master's thesis, Massachusetts Institute of Technology, 1969.

[637] Velthuizen, R. P., Feature extraction with genetic algorithms for fuzzy clustering, Ph. D. dissertation, University of South Florida, 1996.

[638] Venetsanopoulos, A. N. and I. Singh, Topological optimization of communication networks subject to reliability constraints, *Problems of Control and Information Theory*, vol. 15, pp. 63-78, 1986.

[639] Venugopal, V. and T. T. Narendran, Cell formation in manufacturing systems through simulated annealing: an experimental evaluation, *European Journal of Operational Research*, vol. 63, no. 3, pp. 409-422, 1992.

[640] Venugopal, V. and T. T. Narendran, A genetic algorithm approach to the machine-component grouping problem with multiple objectives, *Computers and Industrial Engineering*, vol. 22, no. 4, pp. 469-480, 1992.

[641] Vignaux, G. A. and Z. Michalewicz, A genetic algorithm for the linear transportation problem, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, pp. 445-452, 1991.

[642] Vohra, T., D. Chen, J. Chang, and H. Chen, A network approach to cell formation in cellular manufacturing, *Int. Journal of Production Research*, vol. 28, no. 11, pp. 2075-2084, 1990.

[643] Voicu, L. I., Multiresolution aspects in genetic algorithms, Ph. D. dissertation, University of Central Florida, 1998.

[644] Volgenant, A., A Lagrangean approach to the degree-constrained minimum spanning tree problem, *European Journal of Operational Research*, vol. 39, pp. 325-331, 1989.

[645] Walker, W., Using the set-covering problem to assign fire companies to fire houses, *European Journal of Operational Research*, vol. 22, pp. 275-277, 1974.

[646] Wall, M. B., Genetic algorithm for resource-constrained scheduling, Ph. D. dissertation, Massachusetts Institute of Technology, 1996.

[647] Walters, G. A. and D. K. Smith, Evolutionary design algorithm for optimal layout of tree networks, *Engineering Optimization*, vol. 24, pp. 261-281, 1995.

[648] Wang, D., A simulated annealing approach for scheduling fundamental runs of grouped jobs, *Computers and Operations Research*, (submitted).

[649] Wang, D., An inexact approach for linear programming with fuzzy objective and resource, *Fuzzy Sets and Systems*, vol. 24, pp. 261-281, 1995.

[650] Wang, D., Modelling and optimization for a type of fuzzy nonlinear programming problems in manufacture systems, *Proceedings of the IEEE Conference on Decision and Control*, vol. 4, pp. 4401-4405, 1996.

[651] Wang, D., R. Cheng, and M. Gen, Scheduling grouped jobs on single machine with genetic algorithm, in Gen, M. and Y. Tsujimura, editors, *Evolutionary Computations and Intelligent Systems*, Gordon and Breach, New York(forthcoming).

[652] Wang, L., Genetic algorithm-based approach to subtask matching and scheduling in heterogeneous computing environments and a comparative study of parallel genetic algorithms, Ph. D. dissertation, Purdue University, 1997.

[653] Wang, M. Y., Scheduling in flowshops with reentrant flows and pallet requirements, Ph. D. dissertation, University of Toronto, 1995.

[654] Wang P., Matrix genome encoding for genetic algorithms, Ph. D. dissertation, Polytechnic University, 1996.

[655] Wang. P. Y., G. S. Wang, and Z. G. Hu, Speeding up the search process of genetic algorithms by fuzzy logic, in Zimmermann[708], pp. 665-671.

[656] Warburton, A., Approximation of Pareto optima in multiple objective shortest path problems, *Operations Research*, vol. 35, pp. 70-79, 1987.

[657] Webster, S. and K. Baker, Scheduling groups of jobs on a single machine, *Operations Research*, vol. 43, pp. 692-703, 1995.

[658] Wemmerlov, U. and D. J. Johnson, Cellular manufacturing at 46 user plants: implementation experiences and performance improvements, *Int. Journal of Production Research*, vol. 35, no. 1, pp. 29-49, 1997.

[659] Wetzel, A., Evaluation of the effectiveness of genetic algorithms in combinatorial optimization, Technical report, University of Pittsburgh, 1983.

[660] Whatley, J. K., editor, *SAS/OR User's Guide: Version 5. Netflow Procedure*. SAS Institute, Inc., Cary, NC, 1985.

[661] White, D. W., GANNet: a genetic algorithm for searching topology and weight spaces in neural

network design-the first step in finding a neural network solution, Ph. D. dissertation, Univesity of Maryland, 1993.

[662] Whitley, D., GENITOR: a different genetic algorithm, in *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, 1989.

[663] Whitley, D., V. Gordan, and K. Mathias, Lamarckian evolution, the Baldwin effect and function optimization, in Davidor et al. [142], pp. 6-15.

[664] Wilkov, R. S., Design of computer networks based on a new reliability measure, in *Symposium on Computer-Communications Networks and Teletraffic*, pp. 371-384, 1972.

[665] Wong, H., Performance analysis for genetic algorithms. Ph. D, dissertation, New Jersey Institute of Technology, 1996.

[666] Wozniak, S. J., Knowledge evolution in active database systems via fuzzy constraint management, Ph. D. dissertation, George Mason University, 1998.

[667] Wright, A., Genetic algorithms for real parameter optimization, in Rawlins[512], chapter 4.

[668] Wright, T., Genetic algorithm approach to scheduling resources for a space power system, Ph. D. dissertation, Case Western Reserve University, 1994.

[669] Wu, A. S., Noncoding DNA and floating building blocks for the genetic algorithm, Ph. D. dissertation, University of Michigan, 1995.

[670] Ziao, Y. L., Computer-assisted drug design: genetic algorithms and structures of molecular clusters of aromatic hydrocarbons and actinomycin Ddeoxyguanosine, Ph. D. dissertation, University of Louisville, 1994.

[671] Xie, M. C., Properties and characteristics of genetic algorithms as a problemsolving method, Ph. D. dissertation, Fukui University, 1996.

[672] Xie, Z., Genetic algorithms: the method of regularization and their application to hypocenter location, Ph. D. dissertation, Texas A&M University, 1997.

[673] Xu, H. and G. Vukovich, Fuzzy evolutionary algorithms and automatic robot trajectory generation, in Fogel[197], pp. 595-600.

[674] Xu, Q., Genetic algorithms searching capability and their application for optimization of a model reference fuzzy adaptive controller for linear systems with time delay, Ph. D. dissertation, Musashi Institute of Technology, 1996.

[675] Xu, W., Quadratic minimum spanning tree problems and related topics. Ph. D. dissertation, University of Maryland, 1984.

[676] Yamada, T. and R. Nakano, A genetic algorithm applicable to large-scale jobshop problems, in Männer, R. and B. Manderick, editors, *Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature*, pp. 281-290, Elsevier Science Publishers, New York, 1992.

[677] Yang, X. and M. Gen, Evolution program for bicriteria transportation problem, in Gen and Kobayashi[231], pp. 451-454.

[678] Yang, X. H., Study on signature verification using a genetic algorithm method, Ph. D. dissertation, Nagoya University, 1995.

[679] Yao, L., Parameter estimation for nonlinear systems, Ph. D. dissertation, University of Wisconsin-Madison, 1992.

[680] Yazenin, A. V., Fuzzy and stochastic programming, *Fuzzy Sets and Systems*, vol. 22, pp. 171-180, 1987.

[681] Ye, J., Fuzzy modeling of nonlinear systems and eugenics-based genetic algorithms, Ph. D. dissertation, Okayama University, 1995.

[682] Yeh, M. S., J. S. Lin, and W. C. Yeh, A new Monte Carlo method for estimating network reliability, in Gen and Kobayashi[231], pp. 723-726.

[683] Yokota, D., Study on solving system reliability optimization problems with interval data by genetic algorithms, Ph. D. dissertation, Meiji University, 1996(in Japanese).

[684] Yokota, T., M. Gen, and Y. Li, Genetic algorithms for nonlinear mixed integer programming problems and its applications, *Computers and Industrial Engineering*, vol. 30, no. 4, pp. 905-917, 1996.

[685] Yokota, T. and M. Gen, Optimal interval design for system reliability with incomplete FDS by means of improved genetic algorithms, *Electronics and Communications in Japan*, vol. 81, no. 1, pp. 84-94, 1998.

[686] Yu, P., Multiple criteria decision making: five basic concepts, in Nemhauser, G., A. R., Kan, and M. Todd, editors, *Handbooks in Operations Research and Management Science, Optimization*, vol. 1, chapter 10, pp. 663-699, Elsevier Science Publishers, New York, 1989.

[687] Yu, T. Crystal deformation due to a dipping fault in an elastic gravitational layer overlying a viscoelastic gravitational half-space, Ph. D. dissertation, University of Colorado-Boulder, 1995.

[688] Yue, K. K. and D. J. Lilja, Designing multiprocessor scheduling algorithms using a distributed genetic algorithms system, in Dasgupta, D. and Z. Michalawicz, editors, *Evolutionary Algorithms in Engineering Applications*, pp. 207-222, Springer-Verlag, Heidelberg, 1997.

[689] Yunker, J. M., Optimization of simulation models by genetic algorithms: a comparative study, Ph. D. dissertation, Virginia Polytechnic Institute and State University, 1993.

[690] Zadeh, L., Optimality and non-scalar-valued performance criteria, *IEEE Transactions on Automatic Control*, vol. 8, no. 59, 1963.

[691] Zadeh, L., Fuzzy sets as a basis for a theory of possibility, *Fuzzy Sets and Systems*, vol. 1, pp. 3-28, 1978.

[692] Zeng, X. and B. Rabenasolo, A fuzzy logic based design for adaptive genetic algorithms, in Zimmermann[707], pp. 660-664.

[693] Zheng, D., M. Gen, and R. Cheng, Multiobjective optimization using genetic algorithims, *Engineering Valuation and Cost Analysis*, vol. 2, pp. 303-310, 1999.

[694] Zhou, C. G., Study on the convergence of genetic algorithms and its application to structural determination of feed forward neural networks, Ph. D. dissertation, Saitama University, 1997.

[695] Zhou, G., Study on constrained spanning tree problems with genetic algorithms, Ph. D. dissertation, Ashikaga Institute of Technology, 1999.

[696] Zhou, G. and M. Gen, Evolutionary computation on extended minimum spanning tree

problems, Technical report, Intelligent Systems Engg, Lab. , Ashikaga Institute of Technology, 1999.

[697] Zhou, G. and M. Gen, A new tree encoding for the degree-constrained spanning tree problem, *Soft Computing* (forthcoming).

[698] Zhou, G. and M. Gen, The genetic algorithms approach to the multicriteria minimum spanning tree problem, in Kim, J. H. , X. Yao, and T. Furuhashi, editors, *Proceedings of the First Asia-Pacific Conference on Simulated Evolution and Learning*, PP. 387-394, Taejon, 1996.

[699] Zhou, G. and M. Gen, Approach to degree-constrained minimum spanning tree problem using genetic algorithm, *Engineering Design and Automation*, vol. 3, no. 2, pp. 157-165, 1997.

[700] Zhou, G. and M. Gen, Evolutionary computation on multicriteria production process planning problem, in Porto[512], pp. 419-424.

[701] Zhou, G. and M. Gen, A note on genetic algorithm approach to the degreeconstrained spanning tree problems, *Networks*, vol. 30, pp. 105-109, 1997.

[702] Zhou, G. and M. Gen, An effective genetic algorithm approach to the quadratic minimum spanning tree problem, *Computers and Operations Research*, vol. 25, no. 3, pp. 229-247, 1998.

[703] Zhou, G. and M. Gen, Genetic algorithm approach on multi-criteria minimum spanning tree problem, *European Journal of Operational Research*, vol. 114, pp. 141-152, 1999.

[704] Zhuang, Z. , Ergonomic evaluation of assistive devices and manual methods for resident-handling tasks in nursing homes, Ph. D. dissertation, West Virginia University, 1995.

[705] Zimmermann, H. J. , Fuzzy Programming and linear programming with several objective functions, *Fuzzy Sets and Systems*, vol. 1, no. 1, pp. 45-55, 1978.

[706] Zimmerman, H. , Description and optimization of fuzzy system, *International Journal of General System*, vol. 2, pp. 209-215, 1976.

[707] Zimmermann, H. , *Fuzzy Set Theory and Its Applications*, Kluwer-Nijhoff, Norwell, MA, 1985.

[708] Zimmerman, H. , editor, *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany, 1997.

[709] Bierwirth, C. and D. C. Mattfeld, Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, vol. 7, no. 1, pp. 1-17, 1999.

[710] Bilchev, G. , Evolutionary metaphors for the bin packing problem, *Evolutionary Programming*, pp. 333-341, 1995.

[711] Chen, T. , A hybrid intelligent system for process modeling and control using a neural network and a genetic algorithm, Ph. D. dissertation, The University of Iowa, 1997.

[712] Deb, K. and M. Goyal, A flexible optimization procedure for mechanical component design based on genetic adaptive search, *Trans. of the ASME J. of Mechanical Design*, vol. 120, 1998.

[713] Deb, K. Multi-objective genetic algorithms: problem difficulties and construction of test problems. *Evolutionary Computation*, vpl. 7, no. 3, pp. 205-230, 1999.

[714] Gen, M., K, Ida, and Y. Z. Li, Solving bicriteria solid transportation problem with fuzzy numbers by genetic algorithm, *Computer and Industrial Engineering*, vol. 29, pp. 537-543, 1995.

[715] Gen, M, J. O. Choi, Y. Tsujimura, Genetic algorithm for the capacitated plant location problem with single source constraints, *Proc. of 7th European Congress on Intelligent Techniques & Soft Computing*, *Session CD-7*, *Aachen*, *1999*.

[716] Gen, M., Y. Z. Li, and K. Ida, Solving multi-objective transportation problem by spanning tree-based genetic algorithm, *IEICE Trans. Fundamentals*, vol. E82-A, 1999(forthcoming).

[717] Gen, M., Y. Z. Li, and K. Ida, Spanning tree-based genetic algorithm for bicriteria fixed charge transportation problem, *Journal of Japan Society of Fuzzy Theory and Systems*, 2000 (forthcoming).

[718] Gen, M., Y. Tsujimura, and Y. X. Li, Fuzzy assembly line balancing using genetic algorithms, *Computer and Industrial Engineering*, vol. 31, no. 3-4, pp. 631-634, 1996.

[719] Hadj-Alouane, A. B. and J. C. Bean, A genetic algorithm for the multiplechoice integer program, *Oper. Res.*, vol. 45, no. 1, pp. 92-101, 1997.

[720] Hadj-Alouane, A. B. and J. C. Bean, and K. G. Murty, A hybrid genetic/optimization algorithm for a task allocation problem, *Journal of Scheduling*, vol. 2, pp. 189-201, 1999.

[721] Hsieh, Y., T. Chen, and D. Bricker, Genetic algorithms for reliablity design problems, *Microelectronics and Reliability*, vol. 38, no. 10, pp. 1599-1605, 1998.

[722] Hung, Y. F., C. C. Shin, and C. P. Chen, Evolutionary algorithms for production planning problem with setup decisions, *Journal of the Operational Research Society*, vol. 50, pp. 857-866, 1999.

[723] Jim é nez, F. and J. L. Verdegay, An evolutionary algorithm for interval solid transportation problems, *Evolutionary Computation*, vol. 7, no. 1, pp. 103-107, 1999.

[724] Kim, Y. K., Y. J. Kim, and Y. H. Kim, Genetic algirithms for assembly line balancing with various objectives. *Computer and Industrial Engineering*, vol. 30, no. 3, pp. 397-409, 1996.

[725] Levitin, G., A. Lisnianski, H. Ben-Haim, and D. Elmakis, Redundancy optimization for series parallel multistate systems, *IEEE Transaction on Reliability*, vol. 47, no. 2, pp. 165-172, 1998.

[726] Levitin, G., Sh. Mazal-Tov, and D. Elmakis, Genetic algorithm for open-loop distribution system design, *Electric Power Systems Research*, vol. 32, pp. 81-87, 1995.

[727] Özdamar, L., A genetic algorithm approach to a general category project scheduling problem, *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 29, no. 1, pp. 44-59, 1999.

[728] Mühlenbein, H. and D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm I, continuous parameter optimization, Evolutionary Computation, vol. 1, pp. 25-49, 1993.

[729] Painton, K. and J. Campbell, Genetic algorithm in optimization of system reliability, *IEEE Transaction on Reliability*, vol. 44, no. 2, pp. 172-178, 1995.

[730] Ramachandran, V., V. Sivakumar, and K. Sathiyanarayanan, Genetics based redundancy

optimization, *Microelectronics and Reliability*, vol. 37, no. 4, pp. 661-663, 1997.

[731] Sasaki, M. and M. Gen, Bicriteria knapsack problem with GUB structure by hybrid genetic algorithm, *Journal of Japan Society of Fuzzy Theory and Systems*, 1999(forthcoming).

[732] Smith, A. E. and D. W. Coit, Reliability optimization of series-parallel systems using a genetic algorithm, *IEEE Transaction on Reliability*, vol. 45, no. 2, pp. 254-260, 1996.

[733] Taguchi, T., M. Gen, and K. Ida. Genetic algorithm for solving multiobjective nonlinear integer programming, *IEICE Trans. A*, vol. J79-A, no. 6, pp. 1221-1223, 1996, in Japanese.

[734] Taguchi, T., T. Yokota, and M. Gen, Reliability optimal design problem with interval coefficients using hybrid genetic algoithms, *Computer and Industrial Engineering*, vol. 35, pp. 373 376, 1998.

[735] Takeno, T., and G. Yamazaki, Improvement of local area transportation system scheduling using genetic algorithms, *Engg, Gesign and Auto*. vol. 3, no. 2, pp. 191-200, 1997.

[736] Tsujimura, Y. and M. Gen, Evolutionary project scheduing under resource constraint, H. R. Parsaei, M. Gen, Y. Tsujimura, H. R. Leep and J. P. Wong eds, *Proc. of 2nd Inter, Conf. on Engg, Design and Auto.*, no. 308, 1998.

[737] Zhao, L., Y. Tsujimura, and M. Gen, Genetic algorithm for fuzzy clustering with application to manuiacturing cell formation problem, in M. Jamshidi *etal. eds: Intelligent Automanon and Control; Recent Trends in Development and Applications*, vol. 4, pp. 799-804, 1997.

索　　引

B

C

G

H

J